
BYTE
October – December 1987
LISTINGS
SUPPLEMENT
Including the IBM Special issue

Six great reasons to join BIX today

- **Over 140 *microcomputer-related conferences*:**
Join only those subjects that interest you and change selections at any time. Take part when it's convenient for you. Share information, opinions and ideas in focused discussions with other BIX users who share your interests. Easy commands and conference digests help you quickly locate important information.

- ***Monthly conference specials*:**
BIX specials connect you with invited experts in leading-edge topics—CD-ROM, MIDI, OS-9 and more. They're all part of your BIX membership.

- ***Microbytes daily*:**
Get up-to-the-minute industry news and new product information by joining Microbytes Daily and What's New Hardware and Software.

- ***Public domain software*:**
Yours for the downloading, including programs from BYTE articles and a growing library of PD listings.

- ***Electronic mail*:**
Exchange private messages with BYTE editors and authors and other BIX users.

- ***Vendor support*:**
A growing number of microcomputer manufacturers use BIX to answer your questions about their products and how to use them for peak performance.

What BIX Costs. . How You Pay

ONE-TIME REGISTRATION FEE: $25

Hourly Charges: (Your Time of Access)	**Off-Peak 6PM–7AM** Weekdays Plus Weekends & Holidays	**Peak 7AM–6PM** Weekdays
BIX	$9	$12
Tymnet*	$2	$6
TOTAL	**$11/hr.**	**$18/hr.****

* *Continental U.S. BIX is accessible via Tymnet from throughout the U.S. at charges much less than regular long distance. Call the BIX helpline number listed below for the Tymnet number near you or Tymnet at 1-800-336-0149*

** *User is billed for time on system (i.e., ½ Hr. Off-Peak w/Tymnet = $5.50 charge.)*

BIX and Tymnet charges billed by Visa or Mastercard only.

BIX HELPLINE
(8:30 AM–11:30 PM Eastern Weekdays)

U.S. (except NH)—1-800-227-BYTE
Elsewhere (603) 924-7681

We'll Send You a BIX *User's Manual and Subscriber Agreement as Soon as We've Processed Your Registration.*
JOIN THE EXCITING WORLD OF BIX TODAY!

JOIN BIX RIGHT NOW:
Set your computer's telecommunications program for full duplex, 8-bit characters, even parity, 1 stop bit OR 7-bit characters, even parity, 1 stop using 300 or 1200 baud.
Call your local Tymnet number and respond as follows:

Tymnet Prompt	**You Enter**
Garble or "terminal identifier"	a
login:	bytenet <CR>
password:	migh <CR>
mhis login:	bix <CR>
BIX Logo—Name:	new <CR>

After you register on-line you're immediately taken to the BIX learn conference and can start using the system right away.

FOREIGN ACCESS:
To access BIX from foreign countries you must have an account with your local Postal Telephone & Telegraph (PTT) company. From your PTT enter 3106000157878. Then enter bix <CR> and new <CR> at the prompts. Call or write us for PTT contact information.

BIX
ONE PHOENIX MILL LANE
PETERBOROUGH, NH 03458
(603) 924-9281

TABLE OF CONTENTS

WELCOME TO BYTE'S QUARTERLY LISTINGS SUPPLEMENT

The BYTE Listings Supplement is produced quarterly as a means of providing interested readers with a printed, source code version of those programs referenced in BYTE articles. It provides a far more extensive look into the techniques of coding and the potentialities of microcomputers than we have space for in each month's BYTE.

Programs contained in this Supplement are referenced by the month the article appeared, the page on which their supporting article begins, and the name of the author who wrote the article.

For those who prefer programs already in electronic format, we have a companion service called Listings on Disk. If you have a modem, listings may be downloaded from the BYTEnet bulletin board and, if you are a member of BIX, the "Listings" area also contains programs referenced in BYTE.

If you live outside of the U.S., we've included the names, addresses and telephone numbers of bulletin boards that get program code from us. You'll find the directory just inside the back cover of this Supplement.

The bulletin boards are updated monthly. Several countries have enough boards that the telephone charges for most callers should be the minimum possible.

INDEX

October

November

December

IBM Special 1987

October

DUMP.DEF is referenced in "The OS/2 Applications Family," by Ray Duncan, BYTE, October, 1987 page 102.

```
------------------------------------------------------------
Module definition file: DUMP.DEF
------------------------------------------------------------

NAME DUMP
PROTMODE
DATA MOVEABLE
CODE MOVEABLE PURE
STACKSIZE 4096

------------------------------------------------------------
MAKE file: DUMP
------------------------------------------------------------

dump.obj : dump.asm
  masm /Zi dump;

dump.exe : dump.obj dump.def dump
  link /CO dump,,,doscalls,dump

------------------------------------------------------------
```

DUMP.C is referenced in "The OS/2 Applications Family," by Ray Duncan, BYTE, October, 1987, page 102.

Here is the C code for the DUMP.C program to demonstrate semaphores, multiple threads, etc. from high level.

```
/*
        DUMP.C          Displays the binary contents of a file in
                        hex and ASCII on the standard output device.

                        Program has been deliberately complicated
                        to demonstrate direct calls from C to
                        operating system, use of multiple threads,
                        and synchronization with semaphores.

        Usage is:       C>DUMP unit:path\filename.ext [ >destination ]

        Compile with:   C>CL /AL /Zi /Gs /F 2000 DUMP.C
*/

#include <stdio.h>
#include <malloc.h>
#include <doscalls.h>

#define REC_SIZE 16                     /* size of file records */
#define STK_SIZE 1024                   /* stack size for threads */

char Buf1[REC_SIZE];                    /* first disk buffer */
unsigned Buf1Len;                       /* amount of data in buffer */

char Buf2[REC_SIZE];                    /* second disk buffer */
unsigned Buf2Len;                       /* amount of data in buffer */

unsigned Handle;                        /* file Handle from DOSOPEN */
long filptr;                            /* file offset in bytes */

unsigned long ExitSem;                  /* semaphore for process exit */
unsigned long Buf1FullSem;              /* semaphores for disk buffer #1 */
unsigned long Buf1EmptySem;
unsigned long Buf2FullSem;              /* semaphores for disk buffer #2 */
unsigned long Buf2EmptySem;
```

continued

```
main(int argc, char *argv[])

    void far DisplayThr();              /* entry point for Display Thread */
    void far DiskThr();                 /* entry point for Disk Thread */

    unsigned DisplayThrID;              /* receives Thread ID */
    unsigned DiskThrID;                 /* receives Thread ID */

    char DisplayThrStk[STK_SIZE];       /* allocate stacks for threads */
    char DiskThrStk[STK_SIZE];
    int action;                         /* receives DOSOPEN result */
    int openflag=0x01;                  /* fail open if file not found */
    int openmode=0x40;                  /* read only, deny none */

    filptr=0L;                          /* initialize file pointer */

    ExitSem=0L;                         /* initialize semaphores */
    Buf1EmptySem=Buf1FullSem=0L;
    Buf2EmptySem=Buf2FullSem=0L;
    DOSSEMSET((long) &ExitSem);
    DOSSEMSET((long) &Buf1FullSem);
    DOSSEMSET((long) &Buf2FullSem);

    if (argc < 2)                       /* check command tail */
    {   fprintf(stderr,"\ndump: missing file name\n");
        exit(1);
    }
                                        /* open file or exit */
    if (DOSOPEN(argv[1],&Handle,&action,0L,0,openflag,openmode,0L))
    {   fprintf(stderr,"\ndump: can't find file %s\n",argv[1]);
        exit(1);
    }
                                        /* create Disk Thread */
    if (DOSCREATETHREAD(DiskThr,&DiskThrID,DiskThrStk+STK_SIZE))
    {   fprintf(stderr,"\ndump: can't create Disk Thread");
        exit(1);
    }
                                        /* Create Display Thread */
    if (DOSCREATETHREAD(DisplayThr,&DisplayThrID,DisplayThrStk+STK_SIZE))
    {   fprintf(stderr,"\ndump: can't create Display Thread");
        exit(1);
    }

    DOSSEMWAIT((long) &ExitSem,-1L);    /* wait for exit signal */

    DOSSUSPENDTHREAD(DiskThrID);        /* suspend other threads */
    DOSSUSPENDTHREAD(DisplayThrID);
    DOSCLOSE(Handle);                   /* close file */
    DOSEXIT(1,0);                       /* terminate all threads */
}

/*
        The Disk Thread reads the disk file, alternating between
        Buf1 and Buf2.  This thread gets terminated externally
        when the other threads see end of file has been reached.
*/
void far DiskThr()
{
    while(1)
    {   DOSREAD(Handle,Buf1,REC_SIZE,&Buf1Len); /* read disk */
        SemFlip(&Buf1EmptySem,&Buf1FullSem);    /* mark buffer 1 full */
        DOSSEMWAIT((long) &Buf2EmptySem,-1L);   /* wait for buffer 2 empty */
        DOSREAD(Handle,Buf2,REC_SIZE,&Buf2Len); /* read disk */
        SemFlip(&Buf2EmptySem,&Buf2FullSem);    /* mark buffer 2 full */
        DOSSEMWAIT((long) &Buf1EmptySem,-1L);   /* wait for buffer 1 empty */
    }
}

/*
        The Display Thread formats and displays the data in the
        disk buffers, alternating between Buf1 and Buf2.
*/
void far DisplayThr()


{
    while(1)
    {   DOSSEMWAIT((long) &Buf1FullSem,-1L);    /* wait for buffer 1 full */
        DumpRec(Buf1,Buf1Len);                  /* format and display it */
        SemFlip(&Buf1FullSem,&Buf1EmptySem);    /* mark buffer 1 empty */
        DOSSEMWAIT((long) &Buf2FullSem,-1L);    /* wait for buffer 2 full */
        DumpRec(Buf2,Buf2Len);                  /* format and display it */
        SemFlip(&Buf2FullSem,&Buf2EmptySem);    /* mark buffer 2 empty */
    }
}
```

```
/*
        Display record in hex and ASCII on standard output.
        Clear exit semaphore and terminate thread if record length=0.
*/
DumpRec(char *buffer,int length)
{
    int i;                              /* index to current record */

    if (length==0)                      /* check if record length = 0 */
    {   DOSSEMCLEAR((long) &ExitSem);   /* yes, signal main thread  */
        DOSEXIT(0,0);                   /* and terminate this thread! */
    }

    if (filptr % 128 == 0)              /* maybe print heading */
        printf("\n\n       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F");

    printf("\n%04lX ",filptr);          /* file offset */

    for (i = 0; i < length; i++)        /* print hex equiv. of each byte */
       printf( " %02X", (unsigned char) buffer[i] );

                                        /* space over if partial record */
    if (length != 16) for(i=0; i<(16-length); i++) printf("   ");

    printf("  ");

    for (i = 0; i < length; i++)        /* print ASCII equiv. of bytes */
        (  if (buffer[i] < 32 || buffer[i] > 126) putchar('.');
           else putchar(buffer[i]);
        )

    filptr += REC_SIZE;                 /* update file offset */
}

/*
        Since there is no operation to wait until a semaphore
        is set, we must maintain two semaphores to control each
        buffer and flip them atomically.
*/
SemFlip(long *sem1, long *sem2)
{   DOSENTERCRITSEC();                  /* block other threads */
    DOSSEMSET((long) sem1);             /* set the first semaphore */
    DOSSEMCLEAR((long) sem2);           /* clear the second semaphore */
    DOSEXITCRITSEC();                   /* unblock other threads */
}
```

DUMP.ASM is referenced in "The OS/2 Applications Family," by Ray Duncan, BYTE, October, 1987 page 102.

DUMP.ASM

```
        name    dump
        page    55,132
        title   DUMP --- Display File Contents
        .286c
;
; DUMP.ASM --- a OS/2 utility to display the contents of a
; file on the standard output in hex and ASCII format.
;
; Copyright (C) 1987 Ray Duncan
;
; Usage:  C>DUMP path\filename.ext  [ >device ]
;
; This program has been intentionally complicated
; to demonstrate the use of multiple threads and semaphores
; in a MASM application.  For a roadmap to what is going
; on in this program, see its counterpart DUMP.C.
;

cr      equ     0dh             ; ASCII carriage return
lf      equ     0ah             ; ASCII line feed
blank   equ     20h             ; ASCII space code
tab     equ     09h             ; ASCII tab code

recsize equ     16              ; size of input file records
stksize equ     2048            ; size of stack for threads

stdout  equ     1               ; handle of standard output device
stderr  equ     2               ; handle of standard error device
```

continued

```
        extrn   DOSOPEN:far     ; references to OS/2 services
        extrn   DOSREAD:far
        extrn   DOSWRITE:far
        extrn   DOSCLOSE:far
        extrn   DOSEXIT:far
        extrn   DOSSEMCLEAR:far
        extrn   DOSSEMSET:far
        extrn   DOSSEMWAIT:far
        extrn   DOSALLOCSEG:far
        extrn   DOSCREATETHREAD:far
        extrn   DOSSUSPENDTHREAD:far
        extrn   DOSENTERCRITSEC:far
        extrn   DOSEXITCRITSEC:far
        extrn   DOSGETENV:far

DGROUP  group   _DATA

_DATA           segment word public 'DATA'

ExitSem         dd      0               ; storage for RAM semaphores
Buf1FullSem     dd      0
Buf1EmptySem    dd      0
Buf2FullSem     dd      0
Buf2EmptySem    dd      0

DisplayThrID    dw      0               ; Display thread ID
DiskThrID       dw      0               ; Disk I/O thread ID

Buf1            db      recsize dup (0) ; disk I/O buffer #1
Buf1Len         dw      0               ; length of buffer #1 data

Buf2            db      recsize dup (0) ; disk I/O buffer #2
Buf2Len         dw      0               ; length of buffer #2 data

fname           db      64 dup (0)      ; ASCIIZ name of input file

fhandle         dw      0               ; handle for input file

filptr          dw      0               ; relative address in file

status          dw      0               ; receives status of DOSOPEN

selector        dw      0               ; receives segment selector
                                        ; from DOSALLOCSEG

                                        ; formatting area for output
output          db      'nnnn',blank,blank
outputa         db      16 dup ('nn',blank)
                db      blank
outputb         db      16 dup (blank),cr,lf
output_len      equ     $-output

heading         db      cr,lf           ; heading for each 128 bytes
                db      7 dup (blank)
                db      '0  1  2  3  4  5  6  7  '
                db      '8  9  A  B  C  D  E  F',cr,lf
heading_len     equ     $-heading

msg1            db      cr,lf
                db      'dump: file not found'
                db      cr,lf
msg1_len        equ     $-msg1

msg2            db      cr,lf
                db      'dump: missing file name'
                db      cr,lf
msg2_len        equ     $-msg2

msg3            db      cr,lf
                db      'dump: memory allocation error'
                db      cr,lf
msg3_len        equ     $-msg3

msg4            db      cr,lf
                db      'dump: create thread failed'
                db      cr,lf
msg4_len        equ     $-msg4

_DATA           ends


_TEXT   segment word public 'CODE'
        assume  cs:_TEXT,ds:DGROUP

dump    proc    far                     ; entry point from OS/2
```

```
        call    argc                    ; is filename present?
        cmp     ax,2
        je      dump1                   ; yes, proceed

        mov     dx,offset msg2          ; missing or illegal filespec,
        mov     cx,msg2_len
        jmp     dump9                   ; print error message and exit.

dump1:                                  ; copy filename to local buffer
        mov     ax,1                    ; get ES:BX = filename
        call    argv
        mov     cx,ax                   ; set CX = length
        mov     di,offset fname         ; DS:DI = local buffer
dump15: mov     al,es:[bx]              ; copy it byte by byte
        mov     [di],al
        inc     bx
        inc     di
        loop    dump15

        push    ds                      ; set ES = DS
        pop     es

dump2:                                  ; now try to open file...
        push    ds                      ; ASCIIZ file name
        push    offset DGROUP:fname
        push    ds                      ; receives handle
        push    offset DGROUP:fhandle
        push    ds                      ; receives handle
        push    offset DGROUP:status
        push    0                       ; file size (ignored)
        push    0
        push    0                       ; file attribute = normal
        push    1                       ; OpenFlag = fail if doesn't exist
        push    40h                     ; OpenMode = deny none,read only
        push    0                       ; DWORD reserved
        push    0
        call    DOSOPEN                 ; transfer to OS/2
        or      ax,ax                   ; test status
        jz      dump3                   ; jump if open succeeded

        mov     dx,offset msg1          ; open of input file failed,
        mov     cx,msg1_len
        jmp     dump9                   ; print error msg and exit.

dump3:                                  ; initialize semaphores
        push    ds
        push    offset DGROUP:ExitSem
        call    DOSSEMSET

        push    ds
        push    offset DGROUP:Buf1FullSem
        call    DOSSEMSET

        push    ds
        push    offset DGROUP:Buf2FullSem
        call    DOSSEMSET

                                        ; allocate Disk Thread stack
        push    stksize                 ; size of stack
        push    ds                      ; receives selector for
        push    offset DGROUP:selector  ;   allocated block
        push    0                       ; 0 = segment not shareable
        call    DOSALLOCSEG             ; transfer to OS/2
        or      ax,ax                   ; test status
        jz      dump5                   ; jump if allocation succeeded

dump4:  mov     dx,offset DGROUP:msg3   ; display message
        mov     cx,msg3_len             ; 'memory allocation error'
        jmp     dump9                   ; and exit

dump5:                                  ; create Disk Thread
        push    cs                      ; thread's entry point
        push    offset _TEXT:DiskThread
        push    ds                      ; receives thread ID
        push    offset DGROUP:DiskThrID
        push    selector                ; thread's stack base
        push    stksize
        call    DOSCREATETHREAD         ; transfer to OS/2
        or      ax,ax                   ; test status
        jz      dump7                   ; jump if create succeeded

dump6:  mov     dx,offset DGROUP:msg4   ; create of thread failed,
        mov     cx,msg4_len             ; display error message
        jmp     dump9                   ; and exit
```

continued

```
dump7:                                  ; allocate Display Thread stack
        push    stksize                 ; size of stack
        push    ds                      ; receives selector for
        push    offset DGROUP:selector  ;   allocated block
        push    0                       ; 0 = segment not shareable
        call    DOSALLOCSEG             ; transfer to OS/2
        or      ax,ax                   ; test status
        jnz     dump4                   ; jump if allocation failed

                                        ; create Display Thread
        push    cs                      ; thread's entry point
        push    offset _TEXT:DisplayThread
        push    ds                      ; receives thread ID
        push    offset DGROUP:DisplayThrID
        push    selector                ; thread's stack base
        push    stksize
        call    DOSCREATETHREAD         ; transfer to OS/2
        or      ax,ax                   ; test status
        jnz     dump6                   ; jump if create failed

        push    ds                      ; now wait on exit semaphore
        push    offset DGROUP:ExitSem   ; (it will be triggered
        push    -1                      ; by routine DumpRec when
        push    -1                      ; end of file is reached)
        call    DOSSEMWAIT              ; transfer to OS/2

        push    DiskThrID               ; suspend Disk Thread
        call    DOSSUSPENDTHREAD        ; transfer to OS/2

        push    DisplayThrID            ; suspend Display Thread
        call    DOSSUSPENDTHREAD        ; transfer to OS/2

        push    fhandle                 ; close the input file
        call    DOSCLOSE                ; transfer to OS/2

        push    1                       ; terminate all threads
        push    0                       ; return code 0 for success
        call    DOSEXIT                 ; final exit to OS/2

dump9:                                  ; print error message...
        push    stderr                  ; standard error device handle
        push    ds                      ; address of message
        push    dx
        push    cx                      ; length of message
        push    ds                      ; receives bytes written
        push    offset DGROUP:status
        call    DOSWRITE                ; transfer to OS/2

        push    1                       ; terminate all threads
        push    1                       ; return code <>0 for error
        call    DOSEXIT                 ; final exit to OS/2

dump    endp


DiskThread proc far                     ; this thread performs
                                        ; the file I/O, alternating
                                        ; between the two buffers

                                        ; fill buffer #1
        push    fhandle                 ; handle for input file
        push    ds                      ; address of buffer #1
        push    offset DGROUP:Buf1
        push    recsize                 ; record length requested
        push    ds                      ; receives bytes read
        push    offset DGROUP: Buf1Len
        call    DOSREAD

                                        ; signal buffer 1 has data
        mov     si,offset DGROUP:Buf1EmptySem
        mov     di,offset DGROUP:Buf1FullSem
        call    SemFlip

        push    ds                      ; wait until buffer 2 empty
        push    offset DGROUP:Buf2EmptySem
        push    -1
        push    -1
        call    DOSSEMWAIT

                                        ; fill buffer #2
        push    fhandle                 ; handle for input file
        push    ds                      ; address of buffer #1
        push    offset DGROUP:Buf2
        push    recsize                 ; record length requested
        push    ds                      ; receives bytes read
```

```
        push    offset DGROUP:Buf2Len
        call    DOSREAD

                                        ; signal buffer 2 has data
        mov     si,offset DGROUP:Buf2EmptySem
        mov     di,offset DGROUP:Buf2FullSem
        call    SemFlip

        push    ds                      ; wait until buffer 1 empty
        push    offset DGROUP:Buf1EmptySem
        push    -1
        push    -1
        call    DOSSEMWAIT

        jmp     DiskThread              ; do it again...

DiskThread endp


DisplayThread proc far                  ; formats and displays disk
                                        ; data, alternating between
                                        ; the two disk buffers

        push    ds                      ; wait until buffer #1 full
        push    offset DGROUP:Buf1FullSem
        push    -1
        push    -1
        call    DOSSEMWAIT

        mov     si,offset DGROUP:Buf1   ; display buffer 1
        mov     cx,Buf1Len
        call    DumpRec

                                        ; signal buffer #1 is emptied
        mov     si,offset DGROUP:Buf1FullSem
        mov     di,offset DGROUP:Buf1EmptySem
        call    SemFlip

        push    ds                      ; wait until buffer #2 full
        push    offset DGROUP:Buf2FullSem
        push    -1
        push    -1
        call    DOSSEMWAIT

        mov     si,offset DGROUP:Buf2   ; display buffer 2
        mov     cx,Buf2Len
        call    DumpRec

                                        ; signal buffer #2 is emptied
        mov     si,offset DGROUP:Buf2FullSem
        mov     di,offset DGROUP:Buf2EmptySem
        call    SemFlip

        jmp     DisplayThread           ; do it again...

DisplayThread endp


SemFlip proc    near                    ; Flip status of two
                                        ; semaphores atomically

        call    DOSENTERCRITSEC         ; protect this code sequence

        push    ds                      ; set semaphore #1
        push    si
        call    DOSSEMSET
        push    ds                      ; clear semaphore #2
        push    di
        call    DOSSEMCLEAR

        call    DOSEXITCRITSEC          ; let other threads run again
        ret

SemFlip endp


DumpRec proc    near                    ; formats and displays
                                        ; contents of buffer
                                        ; DS:SI = buffer, CX = length

        or      cx,cx                   ; anything to format?
        jnz     DumpRec1                ; yes, continue
```

continued

```
        push    ds                      ; no, clear exit semaphore
        push    offset DGROUP:ExitSem   ; (releasing wait condition
        call    DOSSEMCLEAR             ; for main thread)

        push    0                       ; and terminate this thread
        push    0
        call    DOSEXIT

DumpRec1:                               ; time for a heading?
        test    filptr,07fh             ; if 128 byte boundary
        jnz     DumpRec2                ; no,jump

        push    stdout                  ; standard output device handle
        push    ds                      ; address of heading text
        push    offset DGROUP:heading
        push    heading_len             ; length of heading
        push    ds                      ; receives bytes written
        push    offset DGROUP:status
        call    DOSWRITE

DumpRec2:                               ; format record data...
        push    cx                      ; save record length

        mov     di,offset output        ; first clear output area
        mov     cx,output_len-2
        mov     al,blank
        rep stosb

        mov     di,offset output        ; convert current file offset
        mov     ax,filptr               ; to ASCII for output
        call    w2hex

        pop     cx                      ; get back record length
        mov     bx,0                    ; initialize record pointer

DumpRec3:                               ; fetch next byte from buffer
        mov     al,[si+bx]
                                        ; store ASCII version of character
        mov     di,offset outputb       ; calculate output string address
        mov     byte ptr [di+bx]        ; if not alphanumeric
        cmp     al,blank                ; just print a dot.
        jb      DumpRec4                ; jump, not alphanumeric.
        cmp     al,7eh
        ja      DumpRec4                ; jump, not alphanumeric.
        mov     [di+bx],al              ; else store ASCII character.

DumpRec4:                               ; now convert binary byte
                                        ; to hex ASCII equivalent
        mov     di,offset outputa       ; calc. position in output string
        add     di,bx                   ; base addr + (offset*3)
        add     di,bx
        add     di,bx                   ; convert data in AL to hex
        call    b2hex                   ; ASCII and store into output

        inc     bx                      ; bump data pointer and loop
        loop    DumpRec3                ; until entire record converted

                                        ; now display formatted data
        push    stdout                  ; standard output device handle
        push    ds                      ; address of text
        push    offset DGROUP:output
        push    output_len              ; length of text
        push    ds
        push    offset DGROUP:status    ; receives bytes written
        call    DOSWRITE

        add     word ptr filptr,recsize ; update file pointer

        ret                             ; return to caller

DumpRec endp


argc    proc    near                    ; count command line arguments
                                        ; returns count in AX

        enter   4,0                     ; make room for local variables
                                        ; and give them names...
envseg  equ     [bp-2]                  ; environment segment
cmdoffs equ     [bp-4]                  ; command line offset

        push    es                      ; save original ES,BX, and CX
        push    bx
        push    cx
```

```
        push    ss                      ; get selector for environment
        lea     ax,envseg               ; and offset of command line
        push    ax
        push    ss
        lea     ax,cmdoffs
        push    ax
        call    DOSGETENV               ; transfer to OS/2
        or      ax,ax                   ; check operation status
        mov     ax,1                    ; force argc >= 1
        jnz     argc3                   ; inexplicable failure

        mov     es,envseg               ; set ES:BX = command line
        mov     bx,cmdoffs

argc0:  inc     bx                      ; ignore useless first field
        cmp     byte ptr es:[bx],0
        jne     argc0

argc1:  mov     cx,-1                   ; set flag = outside argument

argc2:  inc     bx                      ; point to next character
        cmp     byte ptr es:[bx],0
        je      argc3                   ; exit if null byte
        cmp     byte ptr es:[bx],blank
        je      argc1                   ; outside argument if ASCII blank
        cmp     byte ptr es:[bx],tab
        je      argc1                   ; outside argument if ASCII tab

                                        ; otherwise not blank or tab,
        jcxz    argc2                   ; jump if already inside argument

        inc     ax                      ; else found argument, count it
        not     cx                      ; set flag = inside argument
        jmp     argc2                   ; and look at next character

argc3:  pop     cx                      ; restore original BX, CX, ES
        pop     bx
        pop     es
        leave                           ; discard local variables
        ret                             ; return AX = argument count

argc    endp


argv    proc    near                    ; get address and length
                                        ; of command line arguments
                                        ; call with AX = arg. no.
                                        ; return ES:BX = address of
                                        ; argument string, CX = length

        enter   4,0                     ; make room for local variables

        push    cx                      ; save original CX and DI
        push    di

        push    ax                      ; save argument number

        push    ss                      ; get selector for environment
        lea     ax,envseg               ; and offset of command line
        push    ax
        push    ss
        lea     ax,cmdoffs
        push    ax
        call    DOSGETENV               ; transfer to OS/2
        or      ax,ax                   ; test operation status
        pop     ax                      ; restore argument number
        jnz     argv7                   ; jump if DOSGETENV failed

        mov     es,envseg               ; set ES:BX = command line
        mov     bx,cmdoffs

        or      ax,ax                   ; is requested argument=0?
        jz      argv8                   ; yes, jump to get program name

argv0:  inc     bx                      ; scan off first field
        cmp     byte ptr es:[bx],0
        jne     argv0
```

continued

```
        xor     ah,ah                   ; initialize argument counter

argv1:  mov     cx,-1                   ; set flag = outside argument

argv2:  inc     bx                      ; point to next character
        cmp     byte ptr es:[bx],0
        je      argv7                   ; exit if null byte
        cmp     byte ptr es:[bx],blank
        je      argv1                   ; outside argument if ASCII blank
        cmp     byte ptr es:[bx],tab
        je      argv1                   ; outside argument if ASCII tab

                                        ; if not blank or tab...
        jcxz    argv2                   ; jump if already inside argument

        inc     ah                      ; else count arguments found
        cmp     ah,al                   ; is this the one we need?
        je      argv4                   ; yes, go find its length
        not     cx                      ; no, set flag = inside argument
        jmp     argv2                   ; and look at next character

argv4:                                  ; found desired argument, now
                                        ; determine its length...
        mov     ax,bx                   ; save param. starting address

argv5:  inc     bx                      ; point to next character
        cmp     byte ptr es:[bx],0
        je      argv6                   ; found end if null byte
        cmp     byte ptr es:[bx],blank
        je      argv6                   ; found end if ASCII blank
        cmp     byte ptr es:[bx],tab
        jne     argv5                   ; found end if ASCII tab

argv6:  xchg    bx,ax                   ; set ES:BX = argument address
        sub     ax,bx                   ; and AX = argument length
        jmp     argvx                   ; return to caller

argv7:  xor     ax,ax                   ; set AX = 0, argument not found
        jmp     argvx                   ; return to caller

argv8:                                  ; special handling for argv=0
        xor     di,di                   ; find the program name by
        xor     al,al                   ; first skipping over all the
        mov     cx,-1                   ; environment variables...
        cld
argv9:  repne scasb                     ; scan for double null (can't use
        scasb                           ; (SCASW since might be odd addr.)
        jne     argv9                   ; loop if it was a single null
        mov     bx,di                   ; save program name address
        mov     cx,-1                   ; now find its length...
        repne scasb                     ; scan for another null byte
        not     cx                      ; convert CX to length
        dec     cx
        mov     ax,cx                   ; return length in AX

argvx:                                  ; common exit point
        pop     di                      ; restore original CX and DI
        pop     cx
        leave                           ; discard stack frame
        ret                             ; return to caller

argv    endp


w2hex   proc    near                    ; convert word to hex ASCII
                                        ; call with AX=binary value
                                        ;           DI=addr to store string
                                        ; returns AX, DI destroyed
        push    ax
        mov     al,ah
        call    b2hex                   ; convert upper byte
        pop     ax
        call    b2hex                   ; convert lower byte
        ret                             ; back to caller

w2hex   endp


b2hex   proc    near                    ; convert byte to hex ASCII
                                        ; call with AL=binary value
                                        ;           DI=addr to store string
                                        ; returns   AX, DI destroyed

        push    cx                      ; save CX for later
        sub     ah,ah                   ; clear upper byte
```

```
        mov     cl,16
        div     cl                      ; divide binary data by 16
        call    ascii                   ; the quotient becomes the first
        stosb                           ; ASCII character
        mov     al,ah
        call    ascii                   ; the remainder becomes the
        stosb                           ; second ASCII character
        pop     cx                      ; restore contents of CX
        ret

b2hex   endp


ascii   proc    near                    ; convert value 0-0FH in AL
                                        ; into a "hex ASCII" character
        add     al,'0'
        cmp     al,'9'
        jle     ascii2                  ; jump if in range 0-9,
        add     al,'A'-'9'-1            ; offset it to range A-F,
ascii2: ret                             ; return ASCII char. in AL.

ascii   endp

_TEXT   ends

        end     dump
```

BPSIM.C complements "Back-Propagation," by William P. Jones and Josiah Hoskins, BYTE, October, 1987, page 155.

```
/*
 * title:       bpsim.c
 * author:      Josiah C. Hoskins
 * date:        June 1987
 *
 * purpose:     backpropagation learning rule neural net simulator
 *              for the tabula rasa Little Red Riding Hood example
 *
 * description: Bpsim provides an implementation of a neural network
 *              containing a single hidden layer which uses the
 *              generalized backpropagation delta rule for learning.
 *              A simple user interface is supplied for experimenting
 *              with a neural network solution to the Little Red Riding
 *              Hood example described in the text.
 *
 *              In addition, bpsim contains some useful building blocks
 *              for further experimentation with single layer neural
 *              networks. The data structure which describes the general
 *              processing unit allows one to easily investigate different
 *              activation (output) and/or error functions. The utility
 *              function create_link can be used to create links between
 *              any two units by supplying your own create_in_out_links
 *              function. The flexibility of creating units and links
 *              to your specifications allows one to modify the code
 *              to tune the network architecture to problems of interest.
 *
 *              There are some parameters that perhaps need some
 *              explanation. You will notice that the target values are
 *              either 0.1 or 0.9 (corresponding to the binary values
 *              0 or 1). With the sigmoidal function used in out_f the
 *              weights become very large if 0 and 1 are used as targets.
 *              The ON_TOLERANCE value is used as a criteria for an output
 *              value to be considered "on", i.e., close enough to the
 *              target of 0.9 to be considered 1. The learning_rate and
 *              momentum variables may be changed to vary the rate of
 *              learning, however, in general they each should be less
 *              than 1.0.
 *
 *              Bpsim has been compiled using CI-C86 version 2.30 on an
 *              IBM-PC and the Sun C compiler on a Sun 3/160.
 *
 *              Note to compile and link on U*IX machines use:
 *                      cc -o bpsim bpsim.c -lm
 *
 *              For other machines remember to link in the math library.
 *
 * status:      This program may be freely used, modified, and distributed
 *              except for commercial purposes.
 *
```

continued

```
 * Copyright (c) 1987   Josiah C. Hoskins
 */

#include <math.h>
#include <stdio.h>
#include <ctype.h>
#define BUFSIZ          512

#define FALSE           0
#define TRUE            !FALSE
#define NUM_IN          6       /* number of input units */
#define NUM_HID         3       /* number of hidden units */
#define NUM_OUT         7       /* number of output units */
#define TOTAL           (NUM_IN + NUM_HID + NUM_OUT)
#define BIAS_UID        (TOTAL) /* threshold unit */

/* macros to provide indexes for processing units */
#define IN_UID(X)       (X)
#define HID_UID(X)      (NUM_IN + X)
#define OUT_UID(X)      (NUM_IN + NUM_HID + X)
#define TARGET_INDEX(X) (X - (NUM_IN + NUM_HID))

#define WOLF_PATTERN    0
#define GRANDMA_PATTERN 1
#define WOODCUT_PATTERN 2
#define PATTERNS        3       /* number of input patterns */
#define ERROR_TOLERANCE 0.01
#define ON_TOLERANCE    0.8     /* a unit's output is on if > ON_TOLERENCE */
#define NOTIFY          10      /* iterations per dot notification */
#define DEFAULT_ITER    250

struct unit {                   /* general processing unit */
  int    uid;                   /* integer uniquely identifying each unit */
  char   *label;
  double output;                /* activation level */
  double (*unit_out_f)();       /* note output fcn == activation fcn*/
  double delta;                 /* delta for unit */
  double (*unit_delta_f)();     /* ptr to function to calc delta */
  struct link *inlinks;         /* for propagation */
  struct link *outlinks;        /* for back propagation */
} *pu[TOTAL+1];                 /* one extra for the bias unit */

struct link {                   /* link between two processing units */
  char   *label;
  double weight;                /* connection or link weight */
  double data;                  /* used to hold the change in weights */
  int    from_unit;             /* uid of from unit */
  int    to_unit;               /* uid of to unit */
  struct link *next_inlink;
  struct link *next_outlink;
};

int     iterations = DEFAULT_ITER;
double  learning_rate = 0.2;
double  momentum = 0.9;
double  pattern_err[PATTERNS];

/*
 * Input Patterns
 * (Big Ears, Big Eyes, Big Teeth, Kindly, Wrinkled, Handsome)
 *   unit 0    unit 1     unit 2   unit 3   unit 4    unit 5
 */
double  input_pat[PATTERNS+1][NUM_IN] = {
  {1.0, 1.0, 1.0, 0.0, 0.0, 0.0},       /* Wolf */
  {0.0, 1.0, 0.0, 1.0, 1.0, 0.0},       /* Grandma */
  {1.0, 0.0, 0.0, 1.0, 0.0, 1.0},       /* Woodcutter */
  {0.0, 0.0, 0.0, 0.0, 0.0, 0.0},       /* Used for Recognize Mode */
};

/*
 * Target Patterns
 * (Scream, Run Away, Look for Woodcutter, Approach, Kiss on Cheek,
 *      Offer Food, Flirt with)
 */
double  target_pat[PATTERNS][NUM_OUT] = {
  {0.9, 0.9, 0.9, 0.1, 0.1, 0.1, 0.1},  /* response to Wolf */
  {0.1, 0.1, 0.1, 0.9, 0.9, 0.9, 0.1},  /* response to Grandma */
  {0.1, 0.1, 0.1, 0.9, 0.1, 0.9, 0.9},  /* response to Woodcutter */
};

/*
 * function declarations
 */
```

```
void    print_header();
char    get_command();
double  out_f(), delta_f_out(), delta_f_hid(), random(), pattern_error();


main()
{
  char   ch;
  extern struct unit *pu[];

  print_header();
  create_processing_units(pu);
  create_in_out_links(pu);
  for (;;) {
    ch = get_command("\nEnter Command (Learn, Recognize, Quit) => ");
    switch (ch) {
    case 'l':
    case 'L':
      printf("\n\tLEARN MODE\n\n");
      learn(pu);
      break;
    case 'r':
    case 'R':
      printf("\n\tRECOGNIZE MODE\n\n");
      recognize(pu);
      break;
    case 'q':
    case 'Q':
      exit(1);
      break;
    default:
      fprintf(stderr, "Invalid Command\n");
      break;
    }
  }
}


void
print_header()
{
  printf("%s%s%s",
        "\n\tBPSIM -- Back Propagation Learning Rule Neural Net Simulator\n",
        "\t\t for the tabula rasa Little Red Riding Hood example.\n\n",
        "\t\t Written by Josiah C. Hoskins\n");
}


/*
 * create input, hidden, output units (and threshold or bias unit)
 */
create_processing_units(pu)
struct  unit *pu[];
{
  int   id;                     /* processing unit index */
  struct unit *create_unit();

  for (id = IN_UID(0); id < IN_UID(NUM_IN); id++)
    pu[id] = create_unit(id, "input", 0.0, NULL, 0.0, NULL);
  for (id = HID_UID(0); id < HID_UID(NUM_HID); id++)
    pu[id] = create_unit(id, "hidden", 0.0, out_f, 0.0, delta_f_hid);
  for (id = OUT_UID(0); id < OUT_UID(NUM_OUT); id++)
    pu[id] = create_unit(id, "output", 0.0, out_f, 0.0, delta_f_out);
  pu[BIAS_UID] = create_unit(BIAS_UID, "bias", 1.0, NULL, 0.0, NULL);
}


/*
 * create links - fully connected for each layer
 *                note: the bias unit has one link to ea hid and out unit
 */
create_in_out_links(pu)
struct  unit *pu[];
{
  int   i, j;           /* i == to and j == from unit id's */
  struct link *create_link();

  /* fully connected units */
  for (i = HID_UID(0); i < HID_UID(NUM_HID); i++) { /* links to hidden */
    pu[BIAS_UID]->outlinks =
      pu[i]->inlinks = create_link(pu[i]->inlinks, i,
                                   pu[BIAS_UID]->outlinks, BIAS_UID,
                                   (char *)NULL,
                                   random(), 0.0);
```

continued

```
    for (j = IN_UID(0); j < IN_UID(NUM_IN); j++) /* from input units */
      pu[j]->outlinks =
        pu[i]->inlinks = create_link(pu[i]->inlinks, i, pu[j]->outlinks, j,
                                     (char *)NULL, random(), 0.0);
  }
  for (i = OUT_UID(0); i < OUT_UID(NUM_OUT); i++) {     /* links to output */
    pu[BIAS_UID]->outlinks =
          pu[i]->inlinks = create_link(pu[i]->inlinks, i,
                                       pu[BIAS_UID]->outlinks, BIAS_UID,
                                       (char *)NULL, random(), 0.0);
    for (j = HID_UID(0); j < HID_UID(NUM_HID); j++) /* from hidden units */
      pu[j]->outlinks =
        pu[i]->inlinks = create_link(pu[i]->inlinks, i, pu[j]->outlinks, j,
                                     (char *)NULL, random(), 0.0);
  }
}

/*
 * return a random number bet 0.0 and 1.0
 */
double
random()
{
  return((rand() % 32727) / 32737.0);
}


/*
 * the next two functions are general utility functions to create units
 * and create links
 */
struct unit *
create_unit(uid, label, output, out_f, delta, delta_f)
int  uid;
char *label;
double   output, delta;
double   (*out_f)(), (*delta_f)();
{
  struct unit  *unitptr;

  if (!(unitptr = (struct unit *)malloc(sizeof(struct unit)))) {
    fprintf(stderr, "create_unit: not enough memory\n");
    exit(1);
  }
  /* initialize unit data */
  unitptr->uid = uid;
  unitptr->label = label;
  unitptr->output = output;
  unitptr->unit_out_f = out_f;  /* ptr to output fcn */
  unitptr->delta = delta;
  unitptr->unit_delta_f = delta_f;
  return (unitptr);
}


struct link *
create_link(start_inlist, to_uid, start_outlist, from_uid, label, wt, data)
struct  link *start_inlist, *start_outlist;
int     to_uid, from_uid;
char *  label;
double  wt, data;
{
  struct link  *linkptr;

  if (!(linkptr = (struct link *)malloc(sizeof(struct link)))) {
    fprintf(stderr, "create_link: not enough memory\n");
    exit(1);
  }
  /* initialize link data */
  linkptr->label = label;
  linkptr->from_unit = from_uid;
  linkptr->to_unit = to_uid;
  linkptr->weight = wt;
  linkptr->data = data;
  linkptr->next_inlink = start_inlist;
  linkptr->next_outlink = start_outlist;
  return(linkptr);
}


char
get_command(s)
char    *s;
{
  char  command[BUFSIZ];
```

```
  fputs(s, stdout);
  fflush(stdin); fflush(stdout);
  (void)fgets(command, BUFSIZ, stdin);
  return((command[0]));          /* return 1st letter of command */
}


learn(pu)
struct unit *pu[];
{
  register i, temp;
  char   tempstr[BUFSIZ];
  extern int    iterations;
  extern double learning_rate, momentum;
  static char prompt[] = "Enter # iterations (default is 250) => ";
  static char quote1[] = "Perhaps, Little Red Riding Hood ";
  static char quote2[] = "should do more learning.\n";

  printf(prompt);
  fflush(stdin); fflush(stdout);
  gets(tempstr);
  if (temp = atoi(tempstr))
    iterations = temp;

  printf("\nLearning ");
  for (i = 0; i < iterations; i++) {
    if ((i % NOTIFY) == 0) {
      printf(".");
      fflush(stdout);
    }
    bp_learn(pu, (i == iterations-2 || i == iterations-1 || i == iterations));
  }
  printf(" Done\n\n");
  printf("Error for Wolf pattern = \t%lf\n", pattern_err[0]);
  printf("Error for Grandma pattern = \t%lf\n", pattern_err[1]);
  printf("Error for Woodcutter pattern = \t%lf\n", pattern_err[2]);
  if (pattern_err[WOLF_PATTERN] > ERROR_TOLERANCE) {
    printf("\nI don't know the Wolf very well.\n%s%s", quote1, quote2);
  } else if (pattern_err[GRANDMA_PATTERN] > ERROR_TOLERANCE) {
    printf("\nI don't know Grandma very well.\n%s%s", quote1, quote2);
  } else if (pattern_err[WOODCUT_PATTERN] > ERROR_TOLERANCE) {
    printf("\nI don't know Mr. Woodcutter very well.\n%s%s", quote1, quote2);
  } else {
    printf("\nI feel pretty smart, now.\n");
  }
}


/*
 * back propagation learning
 */

bp_learn(pu, save_error)
struct unit *pu[];
int    save_error;
{
  static int count = 0;
  static int pattern = 0;
  extern double pattern_err[PATTERNS];

  init_input_units(pu, pattern); /* initialize input pattern to learn */
  propagate(pu);                 /* calc outputs to check versus targets */
  if (save_error)
    pattern_err[pattern] = pattern_error(pattern, pu);
  bp_adjust_weights(pattern, pu);
  if (pattern < PATTERNS - 1)
    pattern++;
  else
      pattern = 0;
  count++;
}


/*
 * initialize the input units with a specific input pattern to learn
 */
init_input_units(pu, pattern)
struct unit *pu[];
int    pattern;
{
  int   id;

  for (id = IN_UID(0); id < IN_UID(NUM_IN); id++)
    pu[id]->output = input_pat[pattern][id];
}
```

continued

```
/*
 * calculate the activation level of each unit
 */
propagate(pu)
struct unit *pu[];
{
  int   id;

  for (id = HID_UID(0); id < HID_UID(NUM_HID); id++)
    (*(pu[id]->unit_out_f))(pu[id], pu);
  for (id = OUT_UID(0); id < OUT_UID(NUM_OUT); id++)
    (*(pu[id]->unit_out_f))(pu[id], pu);
}


/*
 * function to calculate the activation or output of units
 */
double
out_f(pu_ptr, pu)
struct unit *pu_ptr, *pu[];
{
  double sum = 0.0, exp();
  struct link *tmp_ptr;
  tmp_ptr = pu_ptr->inlinks;
  while (tmp_ptr) {
    /* sum up (outputs from inlinks times weights on the inlinks) */
    sum += pu[tmp_ptr->from_unit]->output * tmp_ptr->weight;
    tmp_ptr = tmp_ptr->next_inlink;
  }
  pu_ptr->output = 1.0/(1.0 + exp(-sum));
}


/*
 * half of the sum of the squares of the errors of the
 * output versus target values
 */
double
pattern_error(pat_num, pu)
int     pat_num;        /* pattern number */
struct  unit *pu[];
{
  int           i;
  double        temp, sum = 0.0;

  for (i = OUT_UID(0); i < OUT_UID(NUM_OUT); i++) {
    temp = target_pat[pat_num][TARGET_INDEX(i)] - pu[i]->output;
    sum += temp * temp;
  }
  return (sum/2.0);
}


bp_adjust_weights(pat_num, pu)
int     pat_num;        /* pattern number */
struct  unit *pu[];
{
  int           i;              /* processing units id */
  double        temp1, temp2, delta, error_sum;
  struct link   *inlink_ptr, *outlink_ptr;

  /* calc deltas */
  for (i = OUT_UID(0); i < OUT_UID(NUM_OUT); i++) /* for each output unit */
    (*(pu[i]->unit_delta_f))(pu, i, pat_num); /* calc delta */
  for (i = HID_UID(0); i < HID_UID(NUM_HID); i++) /* for each hidden unit */
    (*(pu[i]->unit_delta_f))(pu, i);      /* calc delta */
  /* calculate weights */
  for (i = OUT_UID(0); i < OUT_UID(NUM_OUT); i++) {     /* for output units */
    inlink_ptr = pu[i]->inlinks;
    while (inlink_ptr) {        /* for each inlink to output unit */
      temp1 = learning_rate * pu[i]->delta *
        pu[inlink_ptr->from_unit]->output;
      temp2 = momentum * inlink_ptr->data;
      inlink_ptr->data = temp1 + temp2; /* new delta weight */
      inlink_ptr->weight += inlink_ptr->data;   /* new weight */
      inlink_ptr = inlink_ptr->next_inlink;
    }
  }
  for (i = HID_UID(0); i < HID_UID(NUM_HID); i++) { /* for ea hid unit */
    inlink_ptr = pu[i]->inlinks;
    while (inlink_ptr) {        /* for each inlink to output unit */
      temp1 = learning_rate * pu[i]->delta *
        pu[inlink_ptr->from_unit]->output;
      temp2 = momentum * inlink_ptr->data;
```

```
      inlink_ptr->data = temp1 + temp2; /* new delta weight */
      inlink_ptr->weight += inlink_ptr->data;   /* new weight */
        inlink_ptr = inlink_ptr->next_inlink;
    }
  }
}


/*
 * calculate the delta for an output unit
 */
double
delta_f_out(pu, uid, pat_num)
struct unit *pu[];
int    uid, pat_num;
{
  double        temp1, temp2, delta;

  /* calc deltas */
  temp1 = (target_pat[pat_num][TARGET_INDEX(uid)] - pu[uid]->output);
  temp2 = (1.0 - pu[uid]->output);
  delta = temp1 * pu[uid]->output * temp2; /* calc delta */
  pu[uid]->delta = delta; /* store delta to pass on */
}


/*
 * calculate the delta for a hidden unit
 */
double
delta_f_hid(pu, uid)
struct unit *pu[];
int    uid;
{
  double        temp1, temp2, delta, error_sum;
  struct link   *inlink_ptr, *outlink_ptr;

  outlink_ptr = pu[uid]->outlinks;
  error_sum = 0.0;
  while (outlink_ptr) {
    error_sum += pu[outlink_ptr->to_unit]->delta * outlink_ptr->weight;
    outlink_ptr = outlink_ptr->next_outlink;
  }
  delta = pu[uid]->output * (1.0 - pu[uid]->output) * error_sum;
  pu[uid]->delta = delta;
}


recognize(pu)
struct unit *pu[];
{
  int    i;
  char   tempstr[BUFSIZ];
  static char *p[] = {"Big Ears?", "Big Eyes?", "Big Teeth?",
                      "Kindly?\t", "Wrinkled?", "Handsome?");

  for (i = 0; i < NUM_IN; i++) {
    printf("%s\t(y/n) ", p[i]);
    fflush(stdin); fflush(stdout);
    fgets(tempstr, BUFSIZ, stdin);
    if (tempstr[0] == 'Y' || tempstr[0] == 'y')
      input_pat[PATTERNS][i] = 1.0;
    else
      input_pat[PATTERNS][i] = 0.0;
  }
  init_input_units(pu, PATTERNS);
  propagate(pu);
  print_behaviour(pu);
}


print_behaviour(pu)
struct unit *pu[];
{
  int   id, count = 0;
  static char *behaviour[] = {
    "Screams", "Runs Away", "Looks for Woodcutter", "Approaches",
    "Kisses on Cheek", "Offers Food", "Flirts with Woodcutter" );
```

continued

```
  printf("\nLittle Red Riding Hood: \n");
  for (id = OUT_UID(0); id < OUT_UID(NUM_OUT); id++){ /* links to out units */
    if (pu[id]->output > ON_TOLERANCE)
      printf("\t%s\n", behaviour[count]);
    count++;
  }
  printf("\n");
}
```

SIEVE.CPP accompanies "Advantage C++ and Guidelines C++," by Mark Mallett, BYTE, October, 1987, page 229.

This file contains the two files used for the "sieve" C++ benchmark. You'll have to separate them. They are

sv_cpp.h	The include file used to re-implement all the arithmetic operators.
sieve.cpp	The sieve program, slightly modified to: a) reference the "sv_cpp.h" file b) change the number of loops from 10 to 100.

-------------------- Beginning of sv_cpp.h --------------------

```
/* SV_CPP.H - new "int" class for the seive benchmark for C++ */

class INT {
  int   val;
public:
  INT( int i ) { val = i; }
  INT(){}
/*  ~INT(){} */
  int operator= (int t2);
  int operator+ (int t2);
  int operator- (int t2);
  int operator+= (int t2);
  int operator++ ();
  int operator<= (int t2);

  operator int();

};

/* Include the following statement to prevent inline substitution of
    code implementing arithmetic operations;  enclose it in comments
    to make that code inline.  */
#define inline

/* Likewise, include the following in comments to prevent declaring the
   second term of each operator as a register variable */
/* #define      register */

inline int INT :: operator= ( register int t2 )
{
return( val = t2 );
}

inline int INT :: operator+ ( register int t2 )
{
return( val + t2 );
}

inline int INT :: operator- ( register int t2 )
{
return( val - t2 );
}
inline int INT :: operator+= ( register int t2 )
{
return( val += t2 );
}

inline int INT :: operator++ ( )
{
return( val++ );
}

inline int INT :: operator<= ( register int t2 )
{
return( val <= t2 );
}
```

```
inline int INT :: operator int ( )
{
return( val );
}


/* Now for the trick.. */
#define int     INT

-------------------- End of sv_cpp.h --------------------


-------------------- Beginning of sieve.cpp --------------------

/*
Eratosthenes Sieve Prime Number Program in from BYTE January 1983
*/

#include "sv_cpp.h"

#define TRUE     1
#define FALSE    0
#define size  8190

     char flags [size + 1];
main()
    {
    int i, prime, k, count, iter;

    printf ("100 iterations\n");
    for (iter = 1; iter <= 100; iter++)            /* do program 100 times */
         {
         count = 0;                                /* prime counter */
         for (i = 0; i <= size; i++)               /* set all flags true */
              flags [i] = TRUE;
         for (i = 0; i <= size; i++)
              {
                   if (flags [i])                  /* found a prime */
                   {
                   prime = i + i + 3;              /* twice index + 3 */
/*                 printf ("\n%d", prime);  */
                   for (k = i + prime; k <= size; k+= prime)
                                 flags [k] = FALSE;  /* kill all multiple */
                   count++;                          /* primes found */
                   }
              }
     }
     printf ("\007%d primes.\n", count);     /* primes found on 100th pass */
}

-------------------- End of sieve.cpp --------------------
```

CPLUS.SRC accompanies "Advantage C++ and Guidelines C++," by Mark Mallett, BYTE, October, 1987, page 229.

```
/* score_c.h - Definitions for classes related to the "score" program
               written for the Byte C++ Review

       May 1987 Mark E. Mallett

*/

#ifndef NULL
#define NULL    (char *)0
#endif

/* This header file declares the following classes: */
class   SCORE_ITEM;
class   REST;                           // Derived from SCORE_ITEM
class   NOTE;                           // Derived from SCORE_ITEM
class   CHORD;                          // Derived from SCORE_ITEM

class   SEQUENCE;                       // Relates to SCORE_ITEM
class   SEQ_ITER;                       // Allows following a SEQUENCE

enum NOTEVAL ( C, Csh, D, Dsh, E, F, Fsh, G, Gsh, A, Ash, B );
istream& operator>> ( istream&, NOTEVAL& );
```

continued

```
class SCORE_ITEM                          // Base class for all score items
    {
    int         duration;                 // How long this item persists
    SCORE_ITEM* next;                     // Next SCORE_ITEM in the sequence
    friend class SEQUENCE;
    friend class SEQ_ITER;
    friend istream& operator>> ( istream&, SCORE_ITEM& );
public:
    SCORE_ITEM( int d = 3 ){ duration = d; }
    ~SCORE_ITEM(){}

    int dur() { return duration; }        // Retrieves duration value
    virtual void play();                  // Plays a score item
    };


class REST : public SCORE_ITEM            // Musical rest-- marks a place
    {
public:
    REST( int d = 3 ) : ( d ) {}

    void play();
    };


class NOTE : public SCORE_ITEM
    {
    NOTEVAL     noteval;                  // Value of the note
    friend istream& operator>> ( istream&, NOTE& );
public:
    NOTE( NOTEVAL n = C, int d = 3 ) : ( d ) { noteval = n; }

    ~NOTE(){};

    void play();
    };

class CHORD: public SCORE_ITEM
    {
    int         notemask;                 // Notes that make up this chord
    friend istream& operator>> ( istream&, CHORD& );
public:
    CHORD( int n = 0 , int d = 3 ) : ( d ) { notemask = n; }
    ~CHORD(){}

    void play();
    };


class SEQUENCE                            // Class allowing sequence operations.
    {
    SCORE_ITEM* item;                     // Points to item
    SCORE_ITEM*  last;                    // Points to last item
    SEQUENCE&   append( SCORE_ITEM *si )
        { if ( item == (SCORE_ITEM *)NULL )  item = last = si;
          last->next = si; last = si; si->next = (SCORE_ITEM *)NULL;
          return *this; }
    friend class SEQ_ITER;
public:
    SEQUENCE(){ item = last = (SCORE_ITEM *)NULL; };
    SEQUENCE( SCORE_ITEM& si ) { item = last = &si; }
    ~SEQUENCE(){}
    SEQUENCE& operator= ( SCORE_ITEM& si ) { item = last = &si; return *this; }
    SEQUENCE& operator= ( SEQUENCE& s2 )
       { item = s2.item; last = s2.last; return *this; }
    SEQUENCE& operator+ ( SCORE_ITEM& si ) { return append( &si ); }
    SEQUENCE& operator+= ( SCORE_ITEM& si ) { return append( &si ); }
    SEQUENCE& operator+= ( SCORE_ITEM* si ) { return append( si ); }
    };


class SEQ_ITER                            // To follow a sequence with
    {
    SEQUENCE*   seq;                      // Ptr to song header
    SCORE_ITEM* si;                       // Ptr to current item
public:
    SEQ_ITER( SEQUENCE& sq ){ seq = &sq; si = sq.item; }
    ~SEQ_ITER(){}
    SCORE_ITEM* operator()();             // Retrieve next one
    };
```

```
-----------------------------------------------------------------------

/* score_c.c++ - Support functions for classes related to the "score" program
                 written for the Byte C++ Review

        May 1987 Mark E. Mallett

*/
#include <stream.h>
#include <string.h>
#include "score_c.h"


/* Table of correlations between note names and noteval mnemonics */
static struct { NOTEVAL nval; char* nname; } Nvtbl[13] = {
        { C,    "C" },
        { Csh,  "Csh" },
        { D,    "D" },
        { Dsh,  "Dsh" },
        { E,    "E" },
        { F,    "F" },
        { Fsh,  "Fsh" },
        { G,    "G" },
        { Gsh,  "Gsh" },
        { A,    "A" },
        { Ash,  "Ash" },
        { B,    "B" },
 { 0,    "BAD" }         };


void SCORE_ITEM :: play()               // play for generic score_items
{
cout << "generic item: duration=" << dec(duration,3) << "\n";
}


void REST :: play()
{
cout << "rest:         duration=" << dec(dur(),3) << "\n";
}


void NOTE :: play()
{
        int     i;

for( i = 0; i < 12; ++i )
    if ( noteval == Nvtbl[i].nval )
        break;
cout << "note:         duration=" << dec(dur(),3)
     << "  value: " << Nvtbl[i].nname << "\n";
}


void CHORD :: play()
{
        int     i;
        int     nval;

cout << "chord:        duration=" << dec(dur(),3) << "  value:";
for( nval = 0; nval < 12; ++nval )
    if ( notemask & (1 << nval ) )
        {
        for( i = 0; i < 12; ++i )
            if ( nval == Nvtbl[i].nval )
                break;
        cout << " " << Nvtbl[i].nname;
  }

cout << "\n";
}



SCORE_ITEM* SEQ_ITER :: operator()()
{
SCORE_ITEM*     tsi = si;               // Copy of current ptr
if ( si != (SCORE_ITEM *)NULL ) si = si -> next;
return tsi;
}


istream& operator>> ( istream& s, NOTEVAL& n )
{
```

continued

```
        int     i;
        char    namebuf[100];

s >> (char *)&namebuf[0];                   // Input name
for( i = 0; i < 12; ++i )
    if ( stricmp( namebuf, Nvtbl[i].nname ) == 0 )
        break;
if ( i < 12 )
    n = Nvtbl[i].nval;
else
    cout << "Error: '" << namebuf << "' is not a note name\n";
return s;
}


istream& operator>> ( istream& s, SCORE_ITEM& si )
{
s >> si.duration;
return s;
}


istream& operator>> ( istream& s, NOTE& n )
{
s >> (SCORE_ITEM& ) n;
s >> n.noteval;
return s;
}


istream& operator>> ( istream& s, CHORD& c )
{
        NOTEVAL nval;
        char    ch;

s >> (SCORE_ITEM& ) c;
c.notemask = 0;                             // No notes
for( ; ; )                                  // Input notenames until no comma

    {
    s >> (NOTEVAL &) nval;                  // Get next note
    c.notemask |= (1 << nval);
    s >> ch;                                // Look for semi
    if ( ch == ';' )                        // Quit when found

 break;
    s.putback( ch );                       // Re-use the not-semi
    }
return s;
}


-------------------------------------------------------------------------

/* score.c++ - Test program for the "score" classes,
               written for the Byte C++ Review

        May 1987 Mark E. Mallett

*/

#include <stream.h>
#include <ctype.h>

#include "score_c.h"
void    play( SEQUENCE& );


main()
{
        SEQUENCE song1;
        NOTE    c1(5), c2(10);

c1 = C;
c2 = G;
song1 = c1;
song1 = song1 + c2 + NOTE(Dsh, 7);
song1 += NOTE(Ash);
play( song1 );
cout << "\n\n";


        SEQUENCE song2;
        char    c;
cout << "Enter each element of the song in the following formats\n\n";
cout << "r dd                    to enter a rest\n";
```

```
cout << "n dd note-name          to enter a note\n";
cout << "c dd note-name... ;     to enter a chord\n";
cout << "p                       to play the song\n";
cout << "x                       to finish\n";

for( ; ; )                                  // Loop until done
    {
    cout << "\n> ";                         // Issue prompt
    cin >> c;                               // Get key character
    if ( isupper( c ) )
        c = tolower( c );
    if ( c == 'x' )
        break;

    switch ( c )                            // Process input
        {
        case 'r':                           // rest
     REST* rest = new REST;
            cin >> *rest;
            song2 += rest;
            cout << "rest entered\n";
            break;

        case 'n':                           // note
            NOTE* note = new NOTE;
            cin >> *note;
            song2 += note;
            cout << "note entered\n";
            break;

        case 'c':                           // chord
            CHORD* chord = new CHORD;
            cin >> *chord;
            song2 += chord;
            cout << "chord entered\n";
            break;

        case 'p':                           // Play
            play( song2 );                  // Play it
            break;

        default:                            // Invalid
            cout << "'" << chr(c) << "' isn't valid.\n";
            break;
        }
    }
}


void play( SEQUENCE &song )                 // Routine to play a song
{
SEQ_ITER next_seq(song);                    // Get an iterator for the song
SCORE_ITEM *siptr;
while ( ( siptr = next_seq() ) != (SCORE_ITEM *)NULL )
    siptr->play();
}
```

*** end Score ***

MINIPRES.LST is referenced in, "Mathematical Reasoning," by Leon Sterling, BYTE, October, 1987, page 177.

```
/*  Program 22.1  A program for solving equations

     solve_equation(Equation,Unknown,Solution) :-
        Solution is a solution to the equation Equation
        in the unknown Unknown.
*/
        :- op(40,xfx,\).
        :- op(50,xfx,^).

     solve_equation(A*B=0,X,Solution) :-
        !,
        factorize(A*B,X,Factors\[]),
        remove_duplicates(Factors,Factors1),
        solve_factors(Factors1,X,Solution).

     solve_equation(Equation,X,Solution) :-
        single_occurrence(X,Equation),
        !,
```

```
        position(X,Equation,[Side|Position]),
        maneuver_sides(Side,Equation,Equation1),
        isolate(Position,Equation1,Solution).

     solve_equation(Lhs=Rhs,X,Solution) :-
        is_polynomial(Lhs,X),
        is_polynomial(Rhs,X),
        !,
        polynomial_normal_form(Lhs-Rhs,X,PolyForm),
        solve_polynomial_equation(PolyForm,X,Solution).

     solve_equation(Equation,X,Solution) :-
        offenders(Equation,X,Offenders),
        multiple(Offenders),
           homogenize(Equation,X,Offenders,Equation1,X1),
        solve_equation(Equation1,X1,Solution1),
        solve_equation(Solution1,X,Solution).

/* Program 22.1b        Supporting code for the factorization method

     factorize(Product,Term,Factors) :-
          Factors is a difference-list of factors of Product
          containing Term.
*/

        factorize(A*B,X,Factors\Rest) :-
                !,factorize(A,X,Factors\Factors1),
                factorize(B,X,Factors1\Rest).
     factorize(C,X,[C|Factors]\Factors) :-
        subterm(X,C),  !.
     factorize(C,X,Factors\Factors).

/*   solve_factors(Factors,Unknown,Solution) :-
        Solution is a solution of the equation Factor=0 in
        the Unknown for some Factor in the list of Factors.
*/

     solve_factors([Factor|Factors],X,Solution) :-
        solve_equation(Factor=0,X,Solution).
     solve_factors([Factor|Factors],X,Solution) :-
        solve_factors(Factors,X,Solution).

/* Program 22.1c  Supporting code for the Isolation method  */

     single_occurrence(Subterm,Term) :-
        occurrence(Subterm,Term,1).

     maneuver_sides(1,Lhs = Rhs,Lhs = Rhs) :- !.
     maneuver_sides(2,Lhs = Rhs,Rhs = Lhs) :- !.

     isolate([N|Position],Equation,IsolatedEquation) :-
        isolax(N,Equation,Equation1),
        isolate(Position,Equation1,IsolatedEquation).
     isolate([],Equation,Equation).

     /* Axioms for Isolation    */

isolax(1,-Lhs = Rhs,Lhs = -Rhs).                          % Unary minus

isolax(1,Term1+Term2 = Rhs,Term1 = Rhs-Term2).            % Addition
isolax(2,Term1+Term2 = Rhs,Term2 = Rhs-Term1).            % Addition

isolax(1,Term1-Term2 = Rhs,Term1 = Rhs+Term2).            % Subtraction
isolax(2,Term1-Term2 = Rhs,Term2 = Term1-Rhs).            % Subtraction

isolax(1,Term1*Term2 = Rhs,Term1 = Rhs/Term2) :-          % Multiplication
   Term2 \= 0.
isolax(2,Term1*Term2 = Rhs,Term2 = Rhs/Term1) :-          % Multiplication
   Term1 \= 0.

isolax(1,Term1/Term2 = Rhs,Term1 = Rhs*Term2) :-          % Division
   Term2 \= 0.
isolax(2,Term1/Term2 = Rhs,Term2 = Term1/Rhs) :-          % Division
   Rhs \= 0.

isolax(1,Term1^Term2 = Rhs,Term1 = Rhs^(-Term2)).         % Exponentiation $$$ ^
isolax(2,Term1^Term2 = Rhs,Term2 = log(base(Term1),Rhs)). % Exponentiation

isolax(1,sin(U) = V,U = arcsin(V)).                       % Sine
isolax(1,sin(U) = V,U = 180 - arcsin(V)).                 % Sine
isolax(1,cos(U) = V,U = arccos(V)).                       % Cosine
isolax(1,cos(U) = V,U = -arccos(V)).                      % Cosine

/* Program 22.1d   Support code for Polynomial methods  */
```

```
        is_polynomial(X,X) :- !.
        is_polynomial(Term,X) :-
                constant(Term),!.
        is_polynomial(Term1+Term2,X) :-
                !,is_polynomial(Term1,X),
                is_polynomial(Term2,X).
        is_polynomial(Term1-Term2,X) :-
                !,is_polynomial(Term1,X),
                is_polynomial(Term2,X).
        is_polynomial(Term1*Term2,X) :-
                !,is_polynomial(Term1,X),
                is_polynomial(Term2,X).
        is_polynomial(Term1/Term2,X) :-
                !,is_polynomial(Term1,X),
                constant(Term2).
        is_polynomial(Term^N,X) :-
                !,natural_number(N),is_polynomial(Term,X).

        natural_number(N) :- integer(N),N > 0,!.
/*  polynomial_normal_form(Expression,Term,PolyNormalForm) :-
          PolyNormalForm  is the polynomial normal form of the
          Expression, which is a polynomial in Term.
*/

     polynomial_normal_form(Polynomial,X,NormalForm) :-
        polynomial_form(Polynomial,X,PolyForm),
        remove_zero_terms(PolyForm,NormalForm), !.

     polynomial_form(X,X,[(1,1)]).
     polynomial_form(X^N,X,[(1,N)]).
     polynomial_form(Term1+Term2,X,PolyForm) :-
        polynomial_form(Term1,X,PolyForm1),
        polynomial_form(Term2,X,PolyForm2),
        add_polynomials(PolyForm1,PolyForm2,PolyForm).
     polynomial_form(Term1-Term2,X,PolyForm) :-
        polynomial_form(Term1,X,PolyForm1),
        polynomial_form(Term2,X,PolyForm2),
        subtract_polynomials(PolyForm1,PolyForm2,PolyForm).
     polynomial_form(Term1*Term2,X,PolyForm) :-
        polynomial_form(Term1,X,PolyForm1),
        polynomial_form(Term2,X,PolyForm2),
        multiply_polynomials(PolyForm1,PolyForm2,PolyForm).
     polynomial_form(Term^N,X,PolyForm) :- !,
        polynomial_form(Term,X,PolyForm1),
        binomial(PolyForm1,N,PolyForm).
     polynomial_form(Term,X,[(Term,0)]) :-
        free_of(X,Term), !.

   remove_zero_terms([(0,N)|Poly],Poly1) :-
        !, remove_zero_terms(Poly,Poly1).
   remove_zero_terms([(C,N)|Poly],[(C,N)|Poly1]) :-
        C \= 0, !, remove_zero_terms(Poly,Poly1).
   remove_zero_terms([],[]).

   /*  Polynomial manipulation routines         */

/*  add_polynomials(Poly1,Poly2,Poly) :-
        Poly is the sum of Poly1 and Poly2, where
        Poly1, Poly2 and Poly are all in polynomial form.
*/

   add_polynomials([],Poly,Poly) :- !.
   add_polynomials(Poly,[],Poly) :- !.
   add_polynomials([(Ai,Ni)|Poly1],[(Aj,Nj)|Poly2],[(Ai,Ni)|Poly]) :-
        Ni > Nj, !, add_polynomials(Poly1,[(Aj,Nj)|Poly2],Poly).
   add_polynomials([(Ai,Ni)|Poly1],[(Aj,Nj)|Poly2],[(A,Ni)|Poly]) :-
        Ni =:= Nj, !, A is Ai+Aj, add_polynomials(Poly1,Poly2,Poly).
   add_polynomials([(Ai,Ni)|Poly1],[(Aj,Nj)|Poly2],[(Aj,Nj)|Poly]) :-
        Ni < Nj, !, add_polynomials([(Ai,Ni)|Poly1],Poly2,Poly).

/*  subtract_polynomials(Poly1,Poly2,Poly) :-
        Poly is the difference of Poly1 and Poly2, where
        Poly1, Poly2 and Poly are all in polynomial form.
*/

   subtract_polynomials(Poly1,Poly2,Poly) :-
        multiply_single(Poly2,(-1,0),Poly3),
        add_polynomials(Poly1,Poly3,Poly), !.

/*  multiply_single(Poly1,Monomial,Poly) :-
        Poly is the product of Poly1 and Monomial, where
        Poly1, and Poly are in polynomial form, and Monomial
        has the form (C,N) denoting the monomial C*X^N.
*/
```

continued

```
   multiply_single([(C1,N1)|Poly1],(C,N),[(C2,N2)|Poly]) :-
        C2 is C1*C, N2 is N1+N, multiply_single(Poly1,(C,N),Poly).
   multiply_single([],Factor,[]).

/*  multiply_polynomials(Poly1,Poly2,Poly) :-
        Poly  is the product of Poly1 and Poly2, where
        Poly1, Poly2 and Poly are all in polynomial form.
*/

   multiply_polynomials([(C,N)|Poly1],Poly2,Poly) :-
        multiply_single(Poly2,(C,N),Poly3),
        multiply_polynomials(Poly1,Poly2,Poly4),
        add_polynomials(Poly3,Poly4,Poly).
   multiply_polynomials([],P,[]).

   binomial(Poly,1,Poly).

/*   solve_polynomial_equation(Equation,Unknown,Solution) :-
        Solution  is a solution to the polynomial Equation
        in the unknown Unknown.
*/

   solve_polynomial_equation(PolyEquation,X,X = -B/A) :-
        linear(PolyEquation), !,
        pad(PolyEquation,[(A,1),(B,0)]).
   solve_polynomial_equation(PolyEquation,X,Solution) :-
        quadratic(PolyEquation), !,
        pad(PolyEquation,[(A,2),(B,1),(C,0)]),
        discriminant(A,B,C,Discriminant),
        root(X,A,B,C,Discriminant,Solution).

        discriminant(A,B,C,D) :- D is B*B - 4*A*C.

        root(X,A,B,C,0,X= -B/(2*A)).
        root(X,A,B,C,D,X= (-B+sqrt(D))/(2*A)) :- D > 0.
        root(X,A,B,C,D,X= (-B-sqrt(D))/(2*A)) :- D > 0.

   pad([(C,N)|Poly],[(C,N)|Poly1]) :-
        !, pad(Poly,Poly1).
   pad(Poly,[(0,N)|Poly1]) :-
        pad(Poly,Poly1).
   pad([],[]).

   linear([(Coeff,1)|Poly]).    quadratic([(Coeff,2)|Poly]).

/* Program 22.1d Supporting code for Homogenization     */

/*  offenders(Equation,Unknown,Offenders)
   Offenders is the set of offenders of the equation in the Unknown  */

offenders(Equation,X,Offenders) :-
   parse(Equation,X,Offenders1\[]),
   remove_duplicates(Offenders1,Offenders).

/*   homogenize(
                                                    */

homogenize(Equation,X,Offenders,Equation1,X1) :-
   reduced_term(X,Offenders,Type,X1),
   rewrite(Offenders,Type,X1,Substitutions),
   substitute(Equation,Substitutions,Equation1).

reduced_term(X,Offenders,Type,X1) :-
   classify(Offenders,X,Type),
   candidate(Type,Offenders,X,X1).

/*  Heuristics for exponential equations    */

classify(Offenders,X,exponential) :-
   exponential_offenders(Offenders,X).

exponential_offenders([A^B|Offs],X) :-
   free_of(X,A), subterm(X,B), exponential_offenders(Offs,X).
exponential_offenders([],X).

candidate(exponential,Offenders,X,A^X) :-
   base(Offenders,A), polynomial_exponents(Offenders,X).

base([A^B|Offs],A) :- base(Offs,A).
base([],A).

polynomial_exponents([A^B|Offs],X) :-
   is_polynomial(B,X), polynomial_exponents(Offs,X).
polynomial_exponents([],X).

/*   Parsing the equation and making substitutions     */
```

```
   /*  parse(Expression,Term,Offenders)
        Expression is traversed to produce the set of Offenders in Term,
        that is the non-algebraic subterms of Expression containing Term  */

      parse(A+B,X,L1\L2) :-
         !, parse(A,X,L1\L3), parse(B,X,L3\L2).
      parse(A*B,X,L1\L2) :-
         !, parse(A,X,L1\L3), parse(B,X,L3\L2).
      parse(A-B,X,L1\L2) :-
         !, parse(A,X,L1\L3), parse(B,X,L3\L2).
      parse(A=B,X,L1\L2) :-
         !, parse(A,X,L1\L3), parse(B,X,L3\L2).
      parse(A^B,X,L) :-
         integer(B), !, parse(A,X,L).
      parse(A,X,L\L) :-
         free_of(X,A), !.
      parse(A,X,[A|L]\L) :-
         subterm(X,A), !.

/*      substitute(Equation,Substitutions,Equation1) :-
        Equation1 is the result of applying the list of
        Substitutions to Equation.
   */
      substitute(A+B,Subs,NewA+NewB) :-
         !, substitute(A,Subs,NewA), substitute(B,Subs,NewB).
      substitute(A*B,Subs,NewA*NewB) :-
         !, substitute(A,Subs,NewA), substitute(B,Subs,NewB).
      substitute(A-B,Subs,NewA-NewB) :-
         !, substitute(A,Subs,NewA), substitute(B,Subs,NewB).
      substitute(A=B,Subs,NewA=NewB) :-
         !, substitute(A,Subs,NewA), substitute(B,Subs,NewB).
      substitute(A^B,Subs,NewA^B) :-
         integer(B), !, substitute(A,Subs,NewA).
      substitute(A,Subs,B) :-
         member(A=B,Subs), !.
      substitute(A,Subs,A).

      /*  Finding homogenization rewrite rules   */

      rewrite([Off|Offs],Type,X1,[Off=Term|Rewrites]) :-
         homog_axiom(Type,Off,X1,Term),
         rewrite(Offs,Type,X1,Rewrites).
      rewrite([],Type,X,[]).


      /*  Homogenization axioms  */

      homog_axiom(exponential,A^(N*X),A^X,(A^X)^N).
      homog_axiom(exponential,A^(-X),A^X,1/(A^X)).
      homog_axiom(exponential,A^(X+B),A^X,A^B*A^X).

/*      Utilities       */

subterm(Term,Term).
subterm(Sub,Term) :-
        compound(Term), functor(Term,F,N), subterm(N,Sub,Term).

subterm(N,Sub,Term) :-
   arg(N,Term,Arg), subterm(Sub,Arg).
subterm(N,Sub,Term) :-
        N > 0,
        N1 is N - 1,
        subterm(N1,Sub,Term).

position(Term,Term,[]) :- !.
position(Sub,Term,Path) :-
        compound(Term), functor(Term,F,N), position(N,Sub,Term,Path), !.

position(N,Sub,Term,[N|Path]) :-
   arg(N,Term,Arg), position(Sub,Arg,Path).
position(N,Sub,Term,Path) :-
   N > 1, N1 is N-1, position(N1,Sub,Term,Path).


      free_of(Subterm,Term) :-
         occurrence(Subterm,Term,N), !, N=0.

      single_occurrence(Subterm,Term) :-
         occurrence(Subterm,Term,N), !, N=1.
         occurrence(Term,Term,1) :- !.
         occurrence(Sub,Term,N) :-
               compound(Term), !, functor(Term,F,M), occurrence(M,Sub,Term,0,N).
         occurrence(Sub,Term,0).
```

continued

```
occurrence(M,Sub,Term,N1,N2) :-
      M > 0, !, arg(M,Term,Arg), occurrence(Sub,Arg,N), N3 is N+N1,
               M1 is M-1, occurrence(M1,Sub,Term,N3,N2).
occurrence(0,Sub,Term,N,N).

multiple([X1,X2|Xs]).
      remove_duplicates([],[]).
      remove_duplicates([X|Xs],[X|Ys]) :-
              remove_duplicates(Xs,Ys).
      remove_duplicates([X|Xs],Ys) :-
              member(X,Xs),
              remove_duplicates(Xs,Ys).
      compound(Term) :- functor(Term,F,N),N > 0,!.

%  Program 22.2      /* Testing and data */
test_press(X,Y) :- equation(X,E,U), solve_equation(E,U,Y).

equation(1,x^2-3*x+2=0,x).

equation(2,cos(x)*(1-2*sin(x))=0,x).

equation(3,2^(2*x) - 5*2^(x+1) + 16 = 0,x).
```

November

ANSISYS.C accompanies, "A C Interface," by Don F. Ridgway,
November, 1987, page 363.

```
/** ansisys.c
*
*   ANSISYS.C
*   (C) Copyright 1985 Don F. Ridgway
*   All rights reserved.
*   This program may be copied for
*   personal, non-profit use only.
*
*   This is an original and unique C programming
*   language header/function file to #include with
*   your C programs to give them "smart" cursor
*   control and eye-catching "turtlegraphics"-
*   type screen and graphics display qualities.
*
*   Programmed by:
*   Don F. Ridgway
*   Owner & Chief Programmer/Analyst
*   A-1 IBM Programming & Training Service
*   CUSTOM BUSINESS PROGRAMS
*   119 Plantation Ct., Suite D
*   Temple Terrace, FL  33617-3731
*   Ph: (813) 985-3342  (10:00 am - 2:00 pm EST)
*
*   Written, compiled & tested in Microsoft C,
*   ver. 2.03, and Lattice C, ver. 2.15, under
*   PC-DOS 2.1 on a Compaq w/640Kb RAM & 8087, using
*   the TURBO Pascal 3.0 screen editor.  (260+ lines.)
*
*   NOTE:  To utilize these macros you must have: (1) The
*   ANSI.SYS file that came with your PC-DOS or MS-DOS 2.xx
*   operating system on your boot disk and, (2) you must boot
*   up with a CONFIG.SYS file on that boot disk which
*   contains the statement: DEVICE = ANSI.SYS.  This loads
*   the ANSI.SYS device driver into DOS at bootup time.  The
*   operating system searches (for) CONFIG.SYS before it looks
*   for an AUTOEXEC.BAT file.  Please refer to your DOS
*   Reference Guide under "ANSI.SYS" and "COPY" for details.
*
*   (Simply, at A> prompt on a boot diskette, type:
*                COPY CON:CONFIG.SYS<cr>
*                DEVICE=ANSI.SYS<cr>
*                <F6><cr>
*   and then reboot and you're ready to go!  Small bother
*   for the brilliant performance gained in your C programs--
*   just have these two files, ANSI.SYS and CONFIG.SYS, on
*   your boot disk whenever you boot up.)
*
*   (The diskette these programs are sent to you on is a
*   PC-DOS 2.1 boot disk so you may boot up with it and run
*   ANSIDEMO.EXE.  Note the ANSI.SYS and CONFIG.SYS files.)
*
*   This custom C module/header file connects the C programming
*   language to the MS-DOS/PC-DOS "ANSI.SYS" device driver
*   used to implement extended screen and keyboard functions.
*   Like any C program, each of the following macros can itself
*   become a building block for a still larger one.  Note the
*   evolution of WINDOW(row1,col1,row2,col2,fill,border) from
*   DRAW(row1,col1,row2,col2,icon) and FILL(row1,col1,row2,col2,fill).
*
*   Please refer to the MS-DOS/PC-DOS Reference Manual and the
*   ANSI.SYS Device Driver commands for the "original" commands
*   and control sequences that are here made into C macros.
*
*   Refer to the IBM Technical Reference Manual or to the
*   appendix of the BASIC Version 2 Reference for the ASCII
*   Character Codes and the Extended Keyboard Function codes.
*
*   Run the ANSIDEMO.EXE for a superb demonstration of all these
*   powerful macros and C programming tools in action.  The
*   actual source code is included in ANSIDEMO.C, an excellent
*   demonstration/introduction to the C programming language.
```

continued

```
*
*   Simply #include "ansisys.c" this file in your programs
*   to enable the following "smart" screen and cursor commands
*   to really supercharge your C programs with professional
*   features that are easier, safer and more portable than
*   tacked-on assembly languge routines.
*
*   Remember that C is "case sensitive" so be sure and
*   reference the following macros with CAPITAL LETTERS.
*
**/

#define BEEP                            printf("\007")
        /* 800 Mz tone for 1/4 second -- same as PRINT CHR$(7) */
#define CLEARSCREEN                     printf("\033[2J")
#define CLS                             CLEARSCREEN
        /* clears the screen and positions cursor at top left corner */
        /* "\033" is Octal for "Escape" or ASCII Decimal 27  (CHR$(27)) */
        /* "Escape-[" is the lead-in for the ANSI.SYS code routines */
#define CURSPOS(x,y)                    printf("\033[%u;%uH",(x),(y))
#define XY(x,y)                         CURSPOS(x,y)
        /* positions cursor at x = row, y = column */
#define EOL                             printf("\033[K")
        /* erases to end of line, including cursor position */
        /* NOTE:  error in DOS documentation has 'K' lower case */
#define XYEOL(x,y)                      printf("\033[%u;%uH\033[K",(x),(y))
        /* positions cursor at x,y then erases to end of line */
#define XYWHERE        printf("\033[6n");scanf("%*1c%2d%*1c%2d%*2c",&row,&col)
        /* requests cursor position, device driver answers row,col--declare int
#define CURSUP(x)                       printf("\033[%uA",(x))
#define CURSDWN(x)                      printf("\033[%uB",(x))
        /* cursor up or down x-many lines */
#define CURSFWD(y)                      printf("\033[%uC",(y))
#define CURSBCK(y)                      printf("\033[%uD",(y))
        /* cursor forward (right) or backward (left) y-many spaces */
#define SAVCURS                         printf("\033[s")
#define RECALLCURS                      printf("\033[u")
        /* cursor position is saved for later recall via RECALLCURS */
#define CPR(x,y,z)                      printf("\033[%u;%uH%c",(x),(y),(z))
#define XYCHAR(x,y,z)                   CPR(x,y,z)
        /* position cursor at x,y and print char z (using ASCII code) */
#define XCTRPRINTF(x,str)   printf("\033[%u;%uH%s",(x),((80-(strlen(str)-1))/2),
        /* on row x, center (and printf) the string str (in double quotes) */
#define CURSPOSPRTF(x,y,str)            printf("\033[%u;%uH%s",(x),(y),str)
#define XYPRINTF(x,y,str)               CURSPOSPRTF(x,y,str)
        /* at position x,y printf the string str (in double quotes) */
#define XKREAD(x)                       x=0;x=bdos(1);if (bdos(11)) x=bdos(8)+128
        /* extended code keyboard read, reads function keys, arrow keys, etc. */
#define XKREADE(x)                      x=0;x=bdos(1);if (bdos(11)) x=bdos(1)+128
        /* same as XKREAD(), except this one echoes the input on the screen */
#define CHKBRK                           if (key==196) break
        /* if F10 key was pressed, break out of loop */
#define SETSCREEN(a)                    printf("\033[=%uh",a)
     /* set screen graphics mode */
     /* 0=40x25 monochrome,1=40x25 color,2=80x25 mono,3=80x25 color,        */
     /* 4=320x200 color,5=320x200 mono,6=640x200 mono,7=enable word-wrap.   */
#define RESETSCREEN(a)                  printf("\033[=%ul",a)
     /* reset screen graphics mode */
     /* the attributes are same as SETSCREEN(a) except 7=disables word-wrap */
#define SETDISPLAY(a,b,c)               printf("\033[%u;%u;%um",a,b,c)
     /* set screen display attributes and colors = (a,b,c) any order:       */
     /* 0 = default, 1 = high intensity, 4 = underline,                     */
/* 5=blinking,7=inverse,8=invisible (black-on-black),30=foreground black,   */
/* 31=fore red,32=fore green,33=fore yellow,34=fore blue,35=fore magenta,   */
/* 36=fore cyan,37=fore white,40=background black,41=back red,42=back green,*/
/* 43=back yellow,44=back blue,45=back magenta,46=back cyan,47=back white.  */
#define HLON                            SETDISPLAY(0,0,1)
        /* set high light (high intensity) on */
#define BLON                            SETDISPLAY(0,0,5)
        /* set blinking on */
#define HLOFF                           SETDISPLAY(0,0,0)
#define BLOFF                           HLOFF
        /* set high intensity, blink (and all other display attributes) to off *
#define PROMPT(x,y,cc)             SETDISPLAY(0,0,7);printf("\033[%u;%uH",(x),(y)
                                        cc=getchar();SETDISPLAY(0,0,0)
        /* at position x,y read inverse prompt for input cc */
#define XKPROMPT(x,y,z)               HLON;XY((x),(y));printf(" \b");XKREAD(z);HLO
        /* at position x,y read highlighted prompt for input z */
#define WINDOW(a,b,c,d,e,f)             DRAW(a,b,c,d,f);FILL(a+1,b+2,c-1,d-2,e)
        /* a rectangle determined by upper left-hand corner coordinates, */
        /* row1 = a, col1 = b, and lower right-hand corner coordinates,  */
        /* row2 = c, col2 = d, is filled with extended graphics character */
        /* ASCII decimal code e, and the border is ASCII decimal code f  */
#define WINDOW2(a,b,c,d,e,f)         DRAW(a,b,c,d,f);DRAW(a+1,b+1,c-1,d-1,255);\
                                        FILL(a+1,b+2,c-1,d-2,e)
```

```
        /* same as WINDOW(a,b,c,d,e,f) except use this one to overwrite other */
        /* drawings because this one fills empty spaces with blanks */

/* ------------------------------------------------------------ */
/** DRAW(row1,col1,row2,col2,icon)                              */
/*                                                              */
/*  can be rectangle, vertical line, horizontal line or point!  */
/*                                                              */
/* row1,col1=Upper Left-hand corner of border                   */
/* row2,col2=Lower Right-hand corner                            */
/* icon=ASCII Decimal number of Character want border made of   */
/*                                                              */
/*   (Note:  Error-trapping is up to you in calling program,    */
/*           e.g., [0<=row<=24], [0<=col<=80], graphics mode,   */
/*           etc.)                                              */
/*                                                              */
/* Dbl Lines=205;Sngl Line=196;Dark=176;Medium=177;Light=178    */
/* White=219;Blank=255;Sunshine=15;Music notes=14;Asterisks=42  */
/* Happy Face=1,2;Hearts=3;Diamonds=4;Clubs=5;Spades=6;Beeps=7  */
/* ------------------------------------------------------------ */
/**/

DRAW(row1,col1,row2,col2,icon)
int row1,col1,row2,col2,icon;
{
   int hlen,vlen,r,c,hzl,vtl,ulc,llc,urc,lrc;

   hlen=col2-col1;
   vlen=row2-row1;
   if (hlen<0 || vlen<0) BEEP;          /* audibly alert possible input error */

   if (hlen<=0 && vlen<=0)              /* then it's a point or a corner */
      {
      CPR(row1,col1,icon);
      return(0);
      }

   if (vlen<=0)                         /* then it's a horizontal line */
      {
      CURSPOS(row1,col1);
      for (c=0;c<=hlen;c++)
         printf("%c",icon);
         return(0);
      }

   switch (icon)
      {
      case 196:                         /* for Single line border */
      case 218:
               hzl=196;vtl=179;ulc=218;llc=192;urc=191;lrc=217;
               break;
      case 201:                         /* for Double line border */
      case 205:
               hzl=205;vtl=186;ulc=201;llc=200;urc=187;lrc=188;
               break;
      case 213:                         /* for Double top, single side */
               hzl=205;vtl=179;ulc=213;llc=212;urc=184;lrc=190;
               break;
      default:
               hzl=vtl=ulc=llc=urc=lrc=icon;  /* for same char all around */
      }

   if (hlen<=0)                 /* it's a vertical line -- use vtl from above */
      {
      CURSPOS(row1,col1);
      for (r=row1;r<=row2;r++)
         CPR(r,col1,vtl);
         return(0);
      }
                                /* if it's fallen through this far it's a rectangle
   CURSPOS(row1,col1);
   for (c=1;c<=hlen;c++)        /* print horizintal icon top row, left to right */
      printf("%c",hzl);
   CPR(row1,col2,urc);          /* print upper right-hand corner */
   for (r=row1+1;r<row2;r++) /* print vertical right-hand column, top to bottom
     CPR(r,col2,vtl);
   CPR(row2,col2,lrc);          /* print lower right-hand corner */
   CURSPOS(row2,col2-1);
   for (c=1;c<=hlen;c++)        /* print horizontal bottom row, right to left */
      printf("%c\b\b",hzl);  /* one forward, two back (NOTE: this is slow) */
   CPR(row2,col1,llc);          /* print lower left-hand corner */
   for (r=row2-1;r>row1;r--) /* print vertical left-hand column, bottom to top *
     CPR(r,col1,vtl);
```

continued

```
   CPR(row1,col1,ulc);        /* print upper left-hand corner to complete object
   return(0);
}                             /* end DRAW() function */

/* ------------------------------------------------------------ */
/** FILL(row1,col1,row2,col2,icon)                              */
/*                                                              */
/*  can be "window," vertical line, horizontal line or point!   */
/*                                                              */
/* row1,col1=Upper Left-hand corner of area-to-be-filled        */
/* row2,col2=Lower Right-hand corner                            */
/* icon=ASCII Decimal number of Character want area filled with */
/*                                                              */
/*   (Note:  Error-trapping is up to you in calling program,    */
/*           e.g., [0<=row<=24], [0<=col<=80], graphics mode,   */
/*           etc.                                               */
/*                                                              */
/* Dbl Lines=205;Sngl Line=196;Dark=176;Medium=177;Light=178    */
/* White=219;Blank=255;Sunshine=15;Music notes=14;Asterisks=42  */
/* Happy Face=1,2;Hearts=3;Diamonds=4;Clubs=5;Spades=6;Beeps=7  */
/* ------------------------------------------------------------ */
/**/

FILL(row1,col1,row2,col2,icon)
int row1,col1,row2,col2,icon;
{
   int hlen,vlen,r,c;

   hlen=col2-col1;
   vlen=row2-row1;
   if (hlen<0 || vlen<0) BEEP;  /* audibly alert possible input error */

   for (r=row1;r<=row2;r++)
     {
     CURSPOS(r,col1);
       {
       for (c=0;c<=hlen;c++)
         printf("%c",icon);
       }
     }
   return(0);
}                             /* end FILL() function */
```

ANSIDEMO.C accompanies, "A C Interface," by Don F. Ridgway, November, 1987, page 363.

```
/** ANSIDEMO.c
*
*   ANSIDEMO.C
*   (C) Copyright 1985 Don F. Ridgway
*   All Rights Reserved.
*   This program may be copied for
*   personal, non-profit use only.
*
*   Don F. Ridgway
*   Owner & Chief Programmer/Analyst
*   A-1 IBM Programming & Training Service
*   Custom Business Programs
*   119 Plantation Court, Suite D
*   Temple Terrace, FL  33617-3731
*   Ph: (813) 985-3342  (10:00am - 2:00pm EST)
*
*   Written, compiled and tested in Microsoft C,
*   ver. 2.03, and Lattice C, ver. 2.15, under
*   PC-DOS 2.1 on a Compaq w/640Kb RAM & 8087
*   using the PC-DOS 3.0 LINK and the TURBO
*   Pascal 3.0 screen editor.
*
*   (470 lines of code.)
*
*   This program demonstrates the features and
*   capabilities of my C programming language
*   header/module file named "ANSISYS.c", which
*   activates and implements the MS/PC-DOS
*   "ANSI.SYS" device driver for extended screen
*   and keyboard functions and control sequences.
*
```

```
*   NOTICE:  To run this program you MUST have booted
*   up with the DOS "ANSI.SYS" file on your boot disk
*   with a "CONFIG.SYS" file containing the statement
*   "device = ansi.sys" on your boot disk.  (Other-
*   wise you'll get meaningless numbers and symbols
*   across your screen.  If so, hit F10 or '0' to exit,
*   then boot up properly.  See the introduction to
*   ANSISYS.C for instructions on how to make the
*   CONFIG.SYS file.)
*
*   The "ANSISYS.c" file is to be #included in
*   your C programs to give them "smart" cursor
*   control and eye-catching "turtlegraphics"-type
*   screen/graphics display capability.
*
**/

#include <stdio.h>
#include "ansisys.c"                    /* this is the file to #include in all
                                        /* your C programs from now on to enable
main()                                  /* all of the following fabulous screen,
   (                                    /* cursor and keyboard features
   int dd,key1;
  while (key1 != 196 && key1 != 48)     /* while key1 not equal to F10 or '0' */
     (                                  /*            show main menu          */
     if (dd==0)
        mainmenu();
     XKPROMPT(20,29,key1);
     dd=0;
     switch(key1)
        (
        case 187:                       /* F1 key -- Set Screen & Graphics    */
        case 49:                        /* '1' key -- just in case some jelly */
                                        /*       spilled on the Function keys */
                 shoscreen();
                 break;
        case 188:                       /* F2 key -- Set Display & Color */
        case 50:                        /* '2' key */
                 shodisplay();
                 break;
        case 189:                       /* F3 key -- Extended Keyboard demo */
        case 51:                        /* '3' key */
                 xkeyboard();
                 break;
        case 190:                       /* F4 key -- Arrow Keys demo */
        case 52:                        /* '4' key */
                 cursarrow();
                 break;
        case 191:                       /* F5 key -- DRAW function demo */
        case 53:                        /* '5' key */
                 showdraw();
                 break;
        case 192:                       /* F6 key -- FILL function demo */
        case 54:                        /* '6' key */
                 showfill();
                 break;
        case 193:                       /* F7 key -- WINDOW function demo */
        case 55:                        /* '7' key */
                 showindow();
                 break;
        case 196:                       /* F10 key -- to exit program */
        case 48:                        /* '0' key */
                 break;
        default:
                 dd=1;                  /* any other key loops back around */
                 break;
        )                               /* end switch */
     )                                  /* end while */

   XYPRINTF(23,1,"Goom\nbye!");
   BEEP;
   exit(0);
   )                                    /* end main ANSIDEMO.c */

/*
* ---------------------------------------------------------------------
*/

mainmenu()                              /* draw Main Menu */
   (
   CLS;                                 /* clear screen */
   DRAW(3,19,23,61,213);                /* draw distinctive one/two line border *
   HLON;                                /* turn high-intensity display on */
   DRAW(4,21,22,59,178);                /* draw artistic inside border to offset
```

continued

```
   XYPRINTF(2,31,"A N S I D E M O . c");
   XYPRINTF(24,29,"(c) 1985 Don F. Ridgway");
   HLOFF;                              /* turn high-intensity off */
   XYPRINTF(6,28,"F1)  Set Screen/Graphics");
   XYPRINTF(8,28,"F2)  Set Display/Color");
   XYPRINTF(10,28,"F3)  Extended Keyboard Keys");
   XYPRINTF(12,28,"F4)  Cursor Arrow Keys");
   XYPRINTF(14,28,"F5)  DRAW Border,Line,Point");
   XYPRINTF(16,28,"F6)  FILL macro/function");
   XYPRINTF(18,28,"F7)  WINDOW macro/function");
   XYPRINTF(20,28,"   <---- (F10 to Exit)");
   return(0);
   )                                   /* return to main program */

/*
* ---------------------------------------------------------------------
*/

shoscreen()                                        /* Set Screen Graphics demo */
   {
   int c;

   CLS;                                            /* clear screen */
   while (c!=9)                                    /* while not number 9 */
      {
      DRAW(1,1,1,80,178);
      XCTRPRINTF(5,"This is Set Screen - Set Graphics Demo");
      XCTRPRINTF(7,"0=40x25 monochrome,1=40x25 color, 2=80x25 mono,  3=80x25 col
      XCTRPRINTF(9,"4=320x200 color,   5=320x200 mono,6=640x200 mono,7=enable wo
      c=' ';
      XYPRINTF(12,5,"Enter # of Graphics Mode desired  ( 9 => Exit) : ");
      scanf("%d",&c);
      if (c==9) break;                             /* if 9 break out of loop */
      SETSCREEN(c);
      XYPRINTF(15,1,"ABCDEFGHIJKLabcdefghijkl");
      XCTRPRINTF(18,"1234567890");
      )

   return(0);
   )                                               /* return to main menu */

/*
* ---------------------------------------------------------------------
*/

shodisplay()                              /* Set Display/Color attributes demo */
   {
   int c,d,e;

   CLS;
   while (c!=9)                                    /* while not number 9 */
      {
      DRAW(1,1,1,80,178);
      XCTRPRINTF(3,"This is Set Display/Color Attributes Demo");
      XCTRPRINTF(5,"Set screen display attributes and colors:");
XYPRINTF(7,1," 0 = default,          1 = high-intensity, 4 = underline,");
XYPRINTF(8,1," 5 = blink,            7 = inverse,        8 = invisible (black-o
XYPRINTF(10,1,"30 = FOREGROUND black, 31 = fore red,      32 = fore green, 33 =
XYPRINTF(11,1,"34 = fore blue,       35 = fore magenta,  36 = fore cyan,  37 =
XYPRINTF(13,1,"40 = BACKGROUND black, 41 = back red,      42 = back green, 43 =
XYPRINTF(14,1,"44 = back blue,       45 = back magenta,  46 = back white.");
      c=d=e=' ';
      XYPRINTF(16,1,"Enter three numbers, seperated by SPACES of Display/Color d
      XYPRINTF(17,1,"putting numbers in right-hand columns first, e.g., 0 0 5 is
      XYPRINTF(18,1,"(A '9'in any column will Exit)  '0 0 0' resets to normal  "
      XY(18,58);
      scanf("%d %d %d",&c,&d,&e);                  /* careful! no error-trapping h
      if (c==9||d==9||e==9) break;                 /* if any number 9, break out *
      SETDISPLAY(c,d,e);
      DRAW(18,58,18,80,255);
      XCTRPRINTF(20,"Is this what you wanted?");
      )

   return(0);
   )                                               /* return to main menu */

/*
* ---------------------------------------------------------------------
*/

xkeyboard()                                        /* Extended Keyboard demo */
   {
   int c;
```

```
   CLS;
   HLON;
   DRAW(1,1,1,80,178);
   HLOFF;
   printf("\nHello there, This is Extended Keyboard Demo ('*'= Exit )\n\n\n");

   while (c!='*')                                  /* while not '*' */
      {
      printf("\n ----> Press ANY key on the keyboard: ");
      XKREAD(c);                                   /* no-echo read */
      printf("  The extended-keyboard-read code =  %d",c);
      }

   return(0);
   }                                               /* return to main menu */

/*
* ---------------------------------------------------------------------
*/

cursarrow()                                /* Display use of cursor arrow keys */
   {
   int key;

   CLS;                                    /* clear screen */
   HLON;
   DRAW(1,1,1,80,178);
   HLOFF;
   XCTRPRINTF(3,"Move cursor with ARROW keys, HOME and END  ('*'= Exit )");
   XY(12,40);
   SAVCURS;                                /* save cursor position (12,40) for later

   while (key != '*')                      /* while not '*' */
      {
      XKREAD(key);                        /* read keystroke, no echo */

      switch(key)
         {
         case 199:                        /* HOME key */
                  XY(1,1);
                  break;
         case 200:                        /* UP arrow key */
                  printf("\033[1A\b");    /* --> NOTE: When utilizing these macros
               /* CURSUP(1); */           /* from the actual keyboard, as we are do
                                          /* in this demo, they need a '\b' backspa
                  break;                  /* because hitting the key moves it forwa
         case 203:                        /* LEFT arrow key */
                  CURSBCK(2);             /* NOTE the (2) per above reason (need tw
                  break;                  /* spaces back to overcome the one forwar
         case 205:                        /* RIGHT arrow */
               /* CURSFWD(1);*/           /* NOTE letting it move forward by itself
                  break;                  /* because of the physical keystroke
         case 207:                        /* END (of screen) key */
                  XY(24,79);
                  break;
         case 208:
                  printf("\033[1B\b");    /* DOWN key */
                /* CURSDWN(1);*/          /* see NOTE on UP arrow key */
                  break;
         case 42:                         /* hit '*' to quit program */
                  break;
         default:                         /* Any other key, while not doing any- */
                  CURSBCK(1);             /* thing, nevertheless needs to be moved
                  break;                  /* back where it was.  See NOTE UP arrow
         )                                /* end switch */
      )                                   /* end while */
      RECALLCURS;                         /* recall cursor (to 12,40) */
      puts("Press any key to return to Main Menu");   /* then print message */
      XKREAD(key);
      CLS;
      return(0);
   )                                      /* end cursarrow demo */

/*
* ---------------------------------------------------------------------
*/

showdraw()                                /* DRAW(row1,col1,row2,col2,icon) demo */
   {
   int a,b,c,d,e,key;
   char *greet;
   greet="Hi there -- I'm Fast-draw Demo!";
```

continued

```
   CLS;
   XCTRPRINTF(2,"Demo of DRAW(row1,col1,row2,col2,icon)");
   DRAW(5,9,20,71,205);                          /* little demo display */
   DRAW(6,11,19,69,176);
   DRAW(7,12,18,68,177);
   DRAW(8,13,17,67,178);
   DRAW(9,14,16,66,219);
   DRAW(11,20,14,60,196);
   HLON;
   XCTRPRINTF(12,greet);
   DRAW(21,3,21,77,207);
   DRAW(22,3,22,77,178);
   DRAW(8,4,8,4,14);
   DRAW(16,4,16,4,2);
   DRAW(8,76,16,76,219);
   DRAW(8,75,16,75,182);
   HLOFF;
   CURSPOSPRTF(24,46,"Press any key to continue ");
   XKREAD(key);
   key=99;
   CLS;
   while (key!=81&&key!=113)     /* while not Capital Q or lower case q q)uit */
                                 /* NOTE: The possibility of upper or lower case
                                 /* entry is handled throughout in this fashion
                                 /* this program could more "stand on its own" a
                                 /* not require the islower() or toupper() funct
                                 /* to be #included from ctype.h header file
      {
      if (key==99||key==67)                      /* if C)learscreen */
         {
         XCTRPRINTF(1,"Demonstration of DRAW(row1,col1,row2,col2,icon)");
         SETDISPLAY(0,0,1);                      /* high intensity */
         DRAW(3,1,3,80,196);                     /* border line    */
         SETDISPLAY(0,0,0);
      CURSPOSPRTF(2,3,"-> Enter row1,col1,row2,col2,icon, SPACES delimiting: ");
 XCTRPRINTF(4,"Try: 10 25 15 55 205, 5 9 20 71 178, 13 1 13 80 219, 9 24 16 56 2
         );
      key=a=b=c=d=e=0;                           /* initialize         */
      CURSPOSPRTF(2,59,".. .. .. .. ...");       /* little input guide -- don't
      CURSPOS(2,59);                             /* to follow exactly but just s
      scanf("%d %d %d %d %d",&a,&b,&c,&d,&e);    /* between entries and hit carr
                                                 /* return at end.  If goof, do
                                                 /* to start all over again. [so
      DRAW(a,b,c,d,e);                           /* Careful! No input error-chec
      CURSPOSPRTF(24,46,"A)nother E)raselast C)LS Q)uit?");
      SETDISPLAY(0,0,5);
      CPR(24,46,65);                             /* blink the   */
      CPR(24,55,69);                             /* A,E,C and Q */
      CPR(24,66,67);
      CPR(24,71,81);
      CURSPOS(24,79);
      SETDISPLAY(0,0,0);
      XKREADE(key);                              /* extended keyboard read */
      if (key==97||key==65)                      /* do A)nother */
         DRAW(24,46,24,80,255);
      else if (key==99||key==67)                 /* C)learScreen,restart */
         CLS;
      else if (key==101||key==69)                /* E)rase last figure */
         {
         DRAW(a,b,c,d,255);                      /* redraw with blanks */
         DRAW(24,46,24,80,255);
         }                                       /* Q)uit falls through */
      }                                          /* end while           */
   CLS;
   return(0);
   }                                             /* end of Showdraw Demo */

/*
* ---------------------------------------------------------------------------
*/


showfill()                              /* Demo of FILL(row1,col1,row2,col2,fill)
   {
   int a,b,c,d,e,key;
   CLS;
   XCTRPRINTF(2,"Demo of FILL(row1,col1,row2,col2,fill)");
   FILL(10,25,15,55,219);                        /* little razzledazzle */
   FILL(5,9,20,71,197);
   FILL(4,40,24,40,221);
   FILL(5,41,20,71,255);
   FILL(10,45,15,75,219);
   CURSPOSPRTF(24,46,"Press any key to continue ");
   XKREAD(key);
```

```
   key=99;
   CLS;
   while (key!=81&&key!=113)                    /* while not Q)uit */
      {
      if (key==99||key==67)                     /* if C)learscreen */
         {
         XCTRPRINTF(1,"Demonstration of FILL(row1,col1,row2,col2,fill)");
         SETDISPLAY(0,0,1);                     /* high intensity */
         DRAW(3,1,3,80,196);                    /* border line    */
         SETDISPLAY(0,0,0);
      CURSPOSPRTF(2,3,"-> Enter row1,col1,row2,col2,fill, SPACES delimiting: ");
XCTRPRINTF(4,"Try: 10 25 15 55 219, 5 9 20 71 197, 4 40 24 40 221, 9 24 16 56 17
         );
      key=a=b=c=d=e=0;                          /* initialize         */
      CURSPOSPRTF(2,59,".. .. .. .. ...");      /* little input guide */
      CURSPOS(2,59);
      scanf("%d %d %d %d %d",&a,&b,&c,&d,&e);
      FILL(a,b,c,d,e);                          /* careful! -- no input */
                                                /* error-checking here! */
      CURSPOSPRTF(24,46,"A)nother E)raselast C)LS Q)uit?");
      SETDISPLAY(0,0,5);
      CPR(24,46,65);                            /* blink the   */
      CPR(24,55,69);                            /* A,E,C and Q */
      CPR(24,66,67);
      CPR(24,71,81);
      CURSPOS(24,79);
      SETDISPLAY(0,0,0);
      XKREADE(key);                             /* extended keyboard read */
      if (key==97||key==65)                     /* do A)nother */
         XYEOL(24,46);
      else if (key==99||key==67)                /* C)learScreen,restart */
         CLS;
      else if (key==101||key==69)               /* E)rase last figure */
         {
         FILL(a,b,c,d,255);                     /* refill with blanks */
         XYEOL(24,46);
         }                                      /* Q)uit falls through */
      }                                         /* end while           */
   CLS;
   return(0);
   }                                            /* end of Showfill Demo */

/*
* ---------------------------------------------------------------------
*/

showindow()                             /* WINDOW(row1,col1,row2,col2,fill,bord)
   {
   int a,b,c,d,e,f,key;
   CLS;                                         /* clearscreen       */
   XCTRPRINTF(2,"Demo of WINDOW(row1,col1,row2,col2,fill,bord)");
   WINDOW(10,25,15,55,219,205);                 /* ----------------- */
   WINDOW(8,23,19,59,178,213);                  /*   if you got it   */
   WINDOW(5,65,10,75,219,196);                  /*   -------------   */
   WINDOW(15,5,20,15,254,205);                  /*     flaunt it!    */
   WINDOW(5,1,7,63,14,219);                     /* ----------------- */
   WINDOW(15,70,15,70,7,2);
   CURSPOSPRTF(24,46,"Press any key to continue ");
   XKREAD(key);
   key=99;
   CLS;
   while (key!=81&&key!=113)                    /* while not Q)uit do  */
      {                                         /* note upper/lowercase */
      if (key==99||key==67)                     /* if screen's cleared */
         {                                      /* redraw header-menu  */
      XCTRPRINTF(1,"Demonstration of WINDOW(row1,col1,row2,col2,fill,bord)");
         SETDISPLAY(0,0,1);                     /* set high intensity  */
         DRAW(3,1,3,80,196);                    /* border line         */
         SETDISPLAY(0,0,0);                     /* set display normal  */
      CURSPOSPRTF(2,1,"-> Enter coordinates & fill & border with SPACE delimit:
XCTRPRINTF(4,"Try: 10 25 15 55 219 205, 5 65 10 75 219 196, 9 24 16 56 219 213")
         );
      key=a=b=c=d=e=f=0;                        /* initialize          */
      CURSPOSPRTF(2,59,".. .. .. .. ... ...");  /* little input guide  */
      CURSPOS(2,59);                            /* position cursor     */
      scanf("%d %d %d %d %d %d",&a,&b,&c,&d,&e,&f); /* BE CAREFUL!!! -- no  */
      WINDOW(a,b,c,d,e,f);                      /* error-checking here! */
      CURSPOSPRTF(24,46,"A)nother E)raselast C)LS Q)uit?");
      SETDISPLAY(0,0,5);                        /* set display to blink */
      CPR(24,46,65);                            /* print/blink the 'A' */
      CPR(24,55,69);                            /* blink 'E'           */
      CPR(24,66,67);                            /* blink 'C'           */
      CPR(24,71,81);                            /* blink 'Q'           */
      CURSPOS(24,79);                           /* position cursor     */
```

```
      SETDISPLAY(0,0,0);                        /* set display normal  */
      XKREADE(key);                             /* extended keyboard read */
      if (key==97||key==65)                     /* do A)nother         */
         XYEOL(24,46);                          /* erase line 24 menu  */
      else if (key==101||key==69)               /* E)rase last figure  */
         {
         FILL(a,b,c,d,255);                     /* refill with blanks  */
         DRAW(24,46,24,80,255);                 /* erase line 24 menu  */
         }
      else if (key==99||key==67)                /* C)learscreen,restart */
         CLS;                                   /* Q)uit falls through */
      }                                         /* end while           */
   CLS;                                         /* clear screen so image */
   return(0);                                   /* ends with program   */
   }                                            /* end of WINDOW Demo */

/*
***  the end of ANSIDEMO.c  --  hope you liked it! --  C you again sometime?  **
*/
```

ARS.C accompanies, "Recursion + Data Structures = Anagrams," by Mike Morton, November, 1987, page 325.

```
/* ARS.C--

   A dictionary-driven program which finds all possible anagrams using a
   recursive search.  Words and phrases are represented by compact "bit
   signatures", which can be easily subtracted with underflow detection.

   Written by Mike Morton in LightspeedC, October 1986.
   Demo version for publication (c) 1986 by Michael S. Morton.
   Portions of the executable version (c) 1986 by Think Technologies,
     Inc.

  Functions included:
    STARTUP:
             choosefields -- decide where fields go in a signature
               makeonesig -- make the bit signature for a word
                   makeuf -- compute the "underflow" signature
                 getwords -- read the dictionary and store all usable
                             words
               storeword -- store a usable word
              makeallsigs -- make bit signatures for all usable words
    SEARCH ROUTINES:
             findanagrams -- recursively search from a given node in the
                             tree
            printanagrams -- print an anagram from the recursion stack
    UTILITY ROUTINES:
                    clean -- remove non-alphabetics and map to lowercase
                   usable -- see if a word is usable in anagrams for a
                             phrase
               makefreqs -- find frequency distribution for a string
              fieldwidth -- compute minimum field width for a frequency
                             count
                     die -- print dying words and cash in the chips
*/

#include "stdio.h"                  /* for terminal and file I/O */
#include "storage.h"                /* for doing mallocs, reallocs */
#include "ctype.h"                  /* for isalpha(), isupper(), etc. */

/**** Easily changeable constants: ****/
#define STRMAX    100               /* size of character strings */
#define STACKMAX  20                /* recursion max (max words/anagram)
                                       */
#define SCRLEN    22                /* count of output lines between
                                       pauses */

/** The anagram-equivalent of a word is stored in a "bit signature"**/

/* NOTE: the "switch" in findanagrams() depends on MAXMASKS */
#define bitmask   long              /* a "bitsig" is made of
                                       "bitmasks" */
#define MAXMASKS  3                 /* at most this many masks per
                                       signature */
typedef bitmask bitsig[MAXMASKS];   /* so, a bit signature looks like
                                       this */
#define maskwidth (8*sizeof(bitmask)) /* number of bits per bitmask */
```

```
/**** Global information about the phrase being anagrammed: ****/
char phrase [STRMAX];       /* the phrase */
int freqs [26];             /* frequency distribution of phrase */
bitsig phrasesig;           /* bit signature for the phrase */
bitsig uflosig;             /* bit signature to detect underflow */

/**** Each letter in the phrase has a field in the bit signature: ****/
int letmask [26];           /* which mask is each letter's field in? */
int letbit [26];            /* what bit # does each field start at? */
int letwidth[26];           /* how wide is field for each letter? */
int lastmask;               /* highest mask # used (0..MAXMASKS-1) */

/**** Dictionary information: ****/
char **wordlist = NULL;     /* dynamic array of pointers to words */
int maxwords = 0;           /* wordlist has bounds [0..maxwords-1] */
int numwords;               /* usable words are in [0..numwords-1] */
char *textnext = NULL;      /* next character to store a word at */
int textleft = 0;           /* characters left in current text chunk */

/* Since we don't always use all bitmasks in a signature, when we
   allocate bit signatures for the usable words, we may use smaller
   signatures (with as few as 1 bitmask).  Hence this array is really
   a list of shrunken bitsigs, but is ACCESSED as bitmasks. */

bitmask *wordsigs;          /* bitsigs for usables; [0..numwords-1] */

/**** For printing anagrams: ****/
char *anawords [STACKMAX];  /* recursion stack to remember words */
char **anaptr;              /* stack ptr (points to 1st unused slot) */
long anacount = 0;          /* total number of anagrams found */

/**** Main program: ****/
/* Get the phrase, neaten it, find the letter frequency distribution.
   Use the distribution to decide where fields go in the bit signature.
   Calculate the signature for the phrase and the "underflow" signature.
   Search the dictionary for all usable words and make their signatures.
   Set up the recursion stack and start the search. */

main()                      /* anagram generator */
{
  printf ("What's the phrase you'd like to anagram? "); /* prompt 'em */
  gets (phrase);            /* get the phrase */
  clean (phrase);           /* remove junk chars; make it lower case */
  makefreqs (phrase, freqs);/* find the frequency distribution */
  choosefields (freqs);     /* assign fields for letters; check size */
  makeuf (freqs, letmask, letbit, letwidth);  /* compute underflow bit
                                                 sig */
  makeonesig (phrase, phrasesig);   /* make bit signature for phrase */

  getwords();               /* get all words usable for this phrase */
  makeallsigs();            /* make signatures for them all */

  anaptr = anawords;                /* initialize the stack pointer */
  findanagrams (0, phrasesig);      /* find & print all anagrams for
                                       phrase */
  printf ("\n   %ld anagrams found.\n", anacount); /* be informative */
}                                   /* end of main program */

/**** clean -- Map alphabetics to lowercase and discard everything
               else. ****/
clean (s)
  char *s;                  /* UPDATE: string to clean in place */
{
  char *out = s;            /* output pointer */

  char c;                   /* working copy of character */

  while (c = *s++)          /* loop through whole input string */
  {
    if (isupper (c)) c -= ('A' - 'a');
                            /* if uppercase, map to lowercase */
    if (isalpha (c)) *out++ = c;    /* and keep only alphabetics */
  }                         /* end of loop mapping & discarding */
  *out++ = c;               /* store the final null */
}                           /* end of clean() */

/**** makefreqs -- Take a phrase and produce its frequency table. ****/
makefreqs (s, ftable)
  char *s;                          /* INPUT: string to analyze */
  int ftable[];                     /* OUTPUT: frequency distribution */
{
  int i;                            /* loop index */
```

continued

```
  for (i = 0; i<26; i++)            /* loop through and initialize... */
    ftable [i] = 0;                 /* ...the frequency array */
  while (*s)                        /* while there's more to the
                                       string... */
    ftable [*s++ - 'a'] ++;         /* ...bump frequency slot; advance
                                          s */
)                                   /* end of makefreqs() */

/**** choosefields -- assign bit positions for each letter in the
                      phrase. ****/
/* For each letter, assign it a field in the bit signature.  To do this,
   find how wide the field is with fieldwidth(), then decide whether
   there's enough room for it in the current mask (kick to the next mask
    if necessary) and allocate room for it.  Then remember all information for
the letter. */

choosefields (freqs)
  int freqs [];                     /* INPUT: phrase's frequency table */
  /* GLOBAL OUTPUT: letmask[], letbit[], letwidth[], lastmask */
{
  int letter;                       /* letter value (0..25) */
  int curmask = 0, curbit = 0;      /* initial mask and bit numbers */
  int width;                        /* fieldwidth of letter's field */

  for (letter = 0; letter < 26; letter++) /* loop through all letters */
    if (freqs[letter] != 0)         /* any occurrences of this letter? */
    {                               /* yes: find where it'll go */
      width = fieldwidth (freqs [letter]); /* how much room does it
                                              need? */

      if (curbit+width > maskwidth) /* too wide to fit in rest of this
                                       mask? */
      {                             /* yes: have to kick into next mask */
        if (++curmask >= MAXMASKS)/* next mask number; is there room? */
          die ("Sorry -- that phrase is too long to handle.\n")
                                    /* nope */
        curbit = 0;                 /* start at 1st bit of next mask */
      }                             /* end of kicking into next mask */

      letmask [letter] = curmask;   /* note which mask this letter goes
                                       in */
      letbit [letter] = curbit;     /* ..and bit position in the mask */
      letwidth [letter] = width;    /* ..and the width */
      curbit += width;              /* advance past this bit field */

    }                           /* end of handling char found in phrase */

  lastmask = curmask;           /* remember highest used mask number */
}                               /* end of choosefields() */

/**** fieldwidth -- Find the width of field needed to store a
                    count. ****/
/* Find how wide a bit field we need to store a small integer.  We're
   passed the maximum count we'll ever see for that letter, and find the
   smallest field which can hold that count.  Then we add in 1 more bit
   for the "underflow" bit.  Our output looks like:
          Freq      Width (+ 1 underflow bit)
          1         1 (+ 1)
          2..3      2 (+ 1)
          4..7      3 (+ 1) ...etc. */

int fieldwidth (count)          /* find width of field to hold "count" */
  int count;                    /* INPUT: frequency of letter */
{
  int width = 1;                /* result -- start at 1 for underflow */

  while (count != 0)            /* loop 'til all bits discarded */
  {
    width++;                    /* counting the bits... */
    count >>= 1;                /* ...and chuck out one more */
  }                             /* end of loop counting bits */
  return (width);               /* that's the answer */
}                               /* end of fieldwidth() */

/**** makeuf -- Make the "underflow" signature. ****/
/* The underflow signature is the only bit signature which doesn't
   correspond to a word or phrase.  Instead, it has the bit just ABOVE
   each letter's count field set.  If too many letters are subtracted
   from a field in a real bit signature, the underflow bit to the left
   of that field will be set, and ANDing the now-bogus signature with
   the "underflow" signature will yield a nonzero result, indicating
   underflow. */
```

```
makeuf (freqs, letmask, letbit, letwidth)
  int freqs [];             /* INPUT: the phrase's frequency table */
  int letmask [], letbit [], letwidth[];/* INPUT: mask #, bit #, field
                                           width */
  /* GLOBAL OUTPUT: uflosig */
{
  int l;                            /* letter number */
  int bnum, bwidth;                 /* bit number, field width */

  for (l = 0; l <= MAXMASKS; l++)   /* to start with, clean out... */
    uflosig [l] = 0;                /* ...each bitmask in the underflow
                                       sig */

  for (l = 0; l < 26; l++)  /* loop through all 26 letters */
    if (freqs [l] != 0)     /* did this letter occur in the phrase? */
    {                       /* yes: it has a field */
      bnum = letbit [l];    /* get the starting bit for the field */
      bwidth = letwidth [l];/* and get the field's width */
      /* Note that we must use "1L", not just "1" -- these are
         longwords. */
      uflosig [letmask [l]] +=  /* take letter's mask from the sig... */
        (1L << (bnum+bwidth-1));/* ...and put the underflow bit in */
    }                           /* end of handling letter in phrase */
}                               /* end of makeuf() */

/**** makeonesig -- Create the bit signature for a string. ****/
makeonesig (str, sig)
  register char *str;               /* INPUT: string to analyze */
  register bitmask sig[];           /* OUTPUT: signature for string */
  /* GLOBAL INPUT: letmask[] and letbit[] */
{
  register int l;           /* letter number (and loop counter) */
  int sfreqs [26];          /* frequency distribution for "str" */
  register bitmask fr;      /* one frequency, shifted into position */

  makefreqs (str, sfreqs);  /* create a frequency table for string */

  for (l = 0; l <= lastmask; l++) /* go through all used bitmasks... */
    sig[l] = 0;                   /* ...initializing their signature */

  for (l = 0; l < 26; l++)          /* loop through all letters */
    if (sfreqs [l])                 /* does this letter occur? */
    {                               /* yes: want to add into its mask */
      fr = ((bitmask) sfreqs [l]) << letbit [l]; /* shift freq ->
                                                    position */
      sig [letmask [l]] += fr;      /* and add into the right mask */
    }                           /* end of adding in letter frequency */
}                               /* end of makeonesig() */

/**** usable -- See if a string is usable; return 0 if not. ****/
int usable (str)                    /* returns zero on failure */
  register char *str;               /* INPUT: string to analyze */
  /* GLOBAL INPUT: frequency table */
{
  register int l;                   /* letter number */
  register char c;                  /* character from the string */
  int sfreqs [26];                  /* string's frequency profile */

  if (*str == '\0') return (0);     /* null string is no good */

  /* We could use makefreqs() here, but we'll usually disqualify the
     word before we build its whole frequency table, so this way is
     faster. */
  for (l = 0; l < 26; l++)          /* loop through all letters... */
    sfreqs [l] = 0;                 /* ...zeroing their frequency */

  while (c = *str++)            /* pick up all characters in str... */
  {
    c -= 'a';                   /* convert each one from a..z to 0..25 */
    if (++sfreqs [c] > freqs[c])    /* tally up their count */
      return (0);               /* ...and see if there are too many */
  }                             /* end of loop through string's letters */

  return (1);                   /* we're OK -- say so */
}                               /* end of usable() */

/**** getwords -- Read in and count all the usable words. ****/
/* The dictionary may be packed: each word is prefixed with an ASCII
   count of the number of letters in common with the previous word.
   Note that if there are no such prefixes (i.e., it's not compressed),
   the routine works fine anyway. */

getwords()
{
```

continued

```
  FILE *dictfile;             /* the dictionary file */
  char inpline [STRMAX];      /* raw input line (still compressed) */
  char lastline [STRMAX];     /* last line read (after unpacking) */
  char *inp;                  /* input line pointer */
  int common;                 /* # of letters in common w/last word */
  char word [STRMAX];         /* word rebuilt from packed dict */
  long wordsread = 0;         /* word count, to give some feedback */

  numwords = 0;                          /* no words yet */
  printf ("Reading dictionary..."); /* explain this unavoidable delay */
  dictfile = fopen ("dictionary", "r"); /* open dictionary;
                                           assume success */
  fseek (dictfile, 0L, 0);  /* reset to start of file */
  while (1)                 /* EOF is noticed in mid-loop */
  {
    if (fgets (inpline, STRMAX, dictfile) == NULL)  /* get a word */
      break;                          /* if end-of-file, quit */
    inpline [strlen(inpline)-1] = '\0'; /* strip off trailing newline */

    common = 0;             /* initially, no letters in common */
    inp = inpline;          /* point to start of input line */
    while (isdigit (*inp))  /* process digits... */
      common = (common * 10) + (*inp++ - '0');
                            /* ...accumulate number */
    lastline [common] = '\0'; /* take first N chars of previous line */
    strcat (lastline, inp);   /* and add the rest to create this word */

    strcpy (word, lastline);  /* remember uncleaned word for next time */
    clean (word);             /* clean up punctuation, etc. */
    if (usable (word) != 0)   /* can it be used in anagrams? */
      storeword (lastline);   /* yes: store the UNcleaned version */
    if ((++wordsread % 1000) == 0) putch ('.');
                              /* whistle while we work */
  }                                 /* end of loop through dict */
  putch ('\n');                     /* finish off the line of dots */

  if (numwords == 0)                /* NOTHING found? */
    die ("Sorry -- absolutely NO usable words found!\n");
  printf ("%d usable words found.\n", numwords);
                                    /* hint at size of output */
}                                   /* end of getwords() */

/**** storeword -- Store a word ****/
/* Because many malloc() routines are slow or space-greedy we batch up
   words and store them in large buffers, allocating buffers
   when needed. */

storeword (word)
  register char *word;              /* word to stash */
{
  register char *memword;           /* allocated word */
  register int len = 1 + strlen (word); /* size we need to store word */
  register int size;                /* size of new chunk */
  /* Decide if the current buffer has enough room for the word. */
  if (len > textleft)             /* no room in current text chunk? */
    if ((textnext = malloc (textleft = 5000))
                                  /* reset size and allocate */
        == NULL)                  /* did we fail? */
      die ("Sorry -- not enough memory for this anagram!\n");

  memword = textnext;         /* allocate at first free character */
  textnext += len;            /* next time, alloc after this word... */
  textleft -= len;            /* ...and debit free length in chunk */
  strcpy (memword, word);     /* store the word in the new chunk */

  /* Store the word's pointer in the dynamic array "wordlist". */
  if ((numwords+1) >= maxwords)
                              /* would this overflow the current list? */
  {
    maxwords += 512;          /* jump to next size */
    size = maxwords * sizeof (char *);
                              /* find new array size, in bytes */

    if (wordlist == NULL)     /* no list yet? */
      wordlist = (char **) malloc (size);
                              /* 1st time: allocate the block */
    else wordlist = (char **) realloc (wordlist, size);
                              /* grow the block */

    if (wordlist == NULL)             /* blew it? */
      die ("Sorry -- not enough memory for this anagram!\n");
  }                                   /* end of handling list overflow */

  wordlist [numwords++] = memword;  /* store this word in the list */
}                                     /* end of storeword() */
```

```
/**** makeallsigs -- Make signatures for each word we saved
                     earlier. ****/
/* Note that "lastmask" may not be as large as MAXMASKS--in other words,
   we don't always use full-sized signatures.  To save memory, we
   allocate them only as large as needed.  Thus some pointers which
   ought to be declared as *bitsig are declared as *bitmask and
   incremented by (lastmask+1) units instead of a simple "++" . */

makeallsigs()
  /* GLOBAL INPUT: numwords, wordlist[] */
  /* GLOBAL OUTPUT: wordsigs[] */
{
  int i;                          /* one more loop index */
  int size;                       /* size of all bit signatures */
  bitmask *sigp;                  /* pointer into array of bitsigs */
  char wordcopy [STRMAX];         /* working copy of a saved word */

  size = numwords * (lastmask+1) * sizeof (bitmask);
                                  /* size of all sigs */
  wordsigs = (bitmask *) malloc (size);/* allocate space for it */
  if (wordsigs == NULL)           /* blew it? */
    die ("Sorry -- not enough memory for this anagram!\n");

  /* Loop through the words; generate and save each one's signature. */
  sigp = wordsigs;                /* point to the first signature
                                     slot */
  for (i = 0; i < numwords; i++)  /* loop through every word... */
  {
    strcpy (wordcopy, wordlist [i]);/* ...copy the word */
    clean (wordcopy);             /* ...clean it up */
    makeonesig (wordcopy, sigp); /* ...make and store its signature */
    sigp += (lastmask+1);         /* ...and bump to next word's slot */
  }                               /* end of loop through usable words */
}                                 /* end of makeallsigs() */

/**** findanagrams -- Recursively search for complete anagrams. ****/
/* We're passed a node, which may be the complete phrase or the
   phrase with some letters already removed.  We're also given the last
   word used, so we can print each permutation in only one order.

   The main loop tries all words from the current word to the last one.
   Each word is "subtracted" from a copy of the current node.  If the
   result underflows, we skip the word.  Otherwise, we push it on our

   stack.  If the result of the subtraction is exactly an empty
   signature (all zero), then we've generated an anagram and can print
   it.  Otherwise we recurse, passing the selected word as the current
   word, and the new, reduced signature as the current node.

   The code to subtract masks is combined with the checks for underflow
   andzero results in a cascaded "switch" which breaks as soon as it
   sees an underflow.  The cases in the switch are nearly identical, and
   are built with the DOMASK macro.  If none of the cases detect
   overflow, the lastcase falls into processing for a successful
   subtraction. */

#define DOMASK(MASK) {          /* one case of switch               */  \
    newmask = curnode [MASK] - cursig [MASK];/*subtract from anagram*/ \
    if (newmask & uflosig [MASK])/* did the subtraction underflow? */  \
        break;                  /* yes: break switch & do next word */ \
    newsig [MASK] = newmask;     /* it's OK; store it                */ \
    bitsleft |= newmask;       /* note if there are any bits left   */ )

findanagrams (curword, curnode)
  register int curword;       /* current word number (used in loop) */
  register bitmask *curnode;  /* bit signature for current node */
{
  bitsig newsig;              /* the new signature (next node down) */
  register bitmask newmask;   /* a single bitmask from new signature */
  register bitmask *cursig;   /* current word's signature */
  register long bitsleft;   /* flag: nonzero if not all letters used */

  cursig = &wordsigs [curword * (lastmask+1)];
                              /* get signature for word */

  while (curword < numwords)  /* loop through all words after this one*/
  {
    bitsleft = 0;             /* no remaining bits seen yet */
    switch (lastmask)         /* check only used masks in signature */
    {
      case 2: DOMASK(2)       /* handle the 3rd mask, if there is one */
      case 1: DOMASK(1)       /* handle the 2nd mask, if there is one */
      case 0: DOMASK(0)       /* handle the 1st mask, if there is one */
```

continued

```
    /* We didn't break, so no underflows occurred.  Print or recurse. */

      *anaptr++ = wordlist [curword];
                                /* stack word for printing/recursing */
   /* Decide whether the anagram is complete or if we must search
      deeper. */
      if (! bitsleft)           /* no bits left in the signature? */
        printanagram();         /* yes: used up all letters!  print it */
      else findanagrams (curword, newsig);
                                /* nope: climb down to subnode */
      --anaptr;                 /* discard the word from the stack */
    )                           /* end of switch to subtract and process */

    curword++;                  /* advance to next word's number... */
    cursig += (lastmask+1); /* ...and to next word's bit signature */
  )                             /* end of loop through words */

)                               /* end of findanagrams() */

/**** printanagram -- Print an anagram stored on the recursion
                      stack. ****/
printanagram()
{
  register char **wp;           /* roving pointer for climbing stack */
  char response [STRMAX];       /* response for when we pause */

  for (wp = anawords; wp < anaptr; wp++)
                                /* go through the whole stack */
    printf ("%s ", *wp);        /* print each word, separated by blanks */
  printf ("\n");                /* kick to a new line after anagram */

  if ((++anacount % SCRLEN) == 0)/* are we at the end of a screenful? */
  {                               /* yes: time to pause */
    printf ("   Press return to continue; anything else to quit...");
    gets (response);                /* see what they want */
    if (strlen (response) != 0) exit();
                                    /* quit if they typed anything */
  )                                 /* end of periodic pause */
)                                   /* end of printanagram() */

/**** die -- Print the passed message and call it quits: ****/
die (s)
  char *s;                          /* dying words */
{
  printf ("%s", s);                 /* help out the user... */
  exit();                           /* and skip town */
)                                   /* end of die() */
```

TM1.BAS accompanies, "The Turing Machine," by Isaac Malitz, November, 1987, page 345

```
1 REM TM1 -- EASY TM SIMULATOR
199 REM TAPE
200 T$="XXXXXbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
210 S$="0":REM STARTING STATE = 0
220 P=1:REM STARTING POSITION ON TAPE = 1
400 PRINT:PRINT T$
410 PRINT TAB(P);"^":REM POSITION OF READ-WRITE HEAD
420 IF S$ = "." THEN END
499 REM L.500 - 520 DOES STATE TABLE LOOKUP
500 M$= S$ + MID$(T$,P,1)
510 RESTORE
520 READ R$:IF MID$(R$,1,2) <> M$ THEN 520
600 PRINT:PRINT S$,R$
610 INPUT "",X
799 REM REWRITE SYMBOL ON TAPE
800 MID$(T$,P,1)=MID$(R$,4,1)
819 REM MOVE LEFT ON TAPE
820 IF MID$(R$,6,1)="L" THEN P=P-1
829 REM MOVE RIGHT ON TAPE
830 IF MID$(R$,6,1)="R" THEN P=P+1
840 S$=MID$(R$,5,1):REM DETERMINE NEW STATE
900 GOTO 400
2000 DATA "0X-b1R","0b-E.." :REM STATE 0
2010 DATA "1X-b0R","1b-O.." :REM STATE 1
```

December

BENCH.PAS Accompanies "Marshal Pascal and Pascal-2," by Mark Bridger, BYTE, December 1987, page 185

```
program SIEVE(INPUT,OUTPUT);
const
  size = 8190;
var
  flags: array[0..size] of boolean;
  I, prime, K, count, iter: integer;
begin
  readln;
  for iter:= 1 to 10 do
    begin
      count:= 0;
      for I:= 0 to size do
      flags[I]:= true;
      for I:= 0 to size do
      if flags[I]
        then
          begin
            prime:= I + I + 3;
            k:= I + prime;
            while K  <= size do
              begin
                flags[K]:= false;
                K:= K + prime;
              end;
            count:= count + 1
          end;
     writeln(count,' primes.');
   end;
 writeln(chr(7)) (Beep)
end.
```

```
program CALC(input,output);
var A,B,C: real;
N, I: integer;
  begin
    readln;
    N:= 5000;
    A:= 2.71828;
    B:= 3.14159;
    C:= 1;
    For I:= 1 to N do
      begin
        C:= C * A;
        C:= C * B;
        C:= C/A;
        C:= C/B
      end;
    write(chr(7));
    writeln('Error = ', C - 1)
  end.
```

```
program FLOATER(INPUT,OUTPUT);
var I: integer;
  x,y: real;
begin
  readln;
  x:= 1;
  for I:= 1 to 1000 do
    begin
      y:= sin(x);
      y:= ln(x);
      y:= exp(x);
      y:= sqrt(x);
      y:= arctan(x);
      x:= x + 0.01
    end;
  write(chr(7));
end.
```

continued

```
{ Turbo Pascal version of trans.pas }
          program TRANS;
          var
            F,G: file of byte;
            ch: byte;
          begin
            readln;
            assign(F, 'infile.txt');
            assign(G, 'outfile.txt');
            reset(F); rewrite(G);
            while not(EOF(F)) do
              begin
                read(F, ch);
                write(G, ch)
              end;
            close(F); close(G);
            write(chr(7));  (Beep)
          end.

{ Standard Pascal version of Trans.pas }
          program TRANS(Input,Output);
          var
            F,G: text;
            ch: char;
          begin
            readln;
            reset(F, 'infile.txt');
            rewrite(G, 'outfile.txt');
            while not(EOF(F)) do
              begin
                read(F, ch);
                write(G, ch)
              end;
            close(F); close(G);
            write(chr(7));  (Beep)
          end.


{ Before running heaptest under Marshal Pascal be sure a specify the large heap

program heaptest(input,output);

const
  AbsoluteMaxSize = 60;

type
  Strings = array[1..9999] of char;
  BufferType = ^Strings;

var
  Buffer: array[0..AbsoluteMaxSize] of BufferType;
  MaxSize, I: integer;


begin
  write('Enter number of pointers: ');
  readln(MaxSize);
  for I:= 1 to MaxSize do new(Buffer[I]);
  dispose(Buffer[MaxSize]);
  for I:= 1 to (MaxSize - 1) div 2 do dispose(Buffer[2*I]);
  new(Buffer[MaxSize]);
  for I:= 1 to (MaxSize - 1) div 2 do new(Buffer[2*I]);
  for I:= 1 to (MaxSize - 1) div 2 do dispose(Buffer[2*I - 1]);
  dispose(Buffer[MaxSize]);
  new(Buffer[0]);
  for I:= 1 to (MaxSize - 1) div 2 do new(Buffer[2*I - 1]);
  write(chr(7));
end.
```

PROJ3D.BAS Accompanies "Three-Dimensional Perspective Plotting," by Tyrone Dault

---

```
REM ************************************************************
REM *                          Three Dimension Plotting Program
REM *                                    Cartesian Coordinates
REM *                                                Version 5.1M
REM * by Tyrone Louis Daulton     October 1986
REM * Washington University in St. Louis
REM *  Department of Physics
REM *
```

```
REM *  Microsoft Basic Verison 2.1
REM *     for Apple Macintosh
REM * This is a streamlined general purpose Cartesian Plotting program.
REM *  This program is fully menu driven.
REM ******************************************************
DIM T(3,3),T1(3,3),T2(3,3)    : REM  Dimension Transformation Arrays
DIM N(3)                                : REM  Dimension Coordinate Array
DIM POLY%(12),PATT%(3)      : REM  Dimension Quick Draw Graphic Arrays
PI=3.141592654#                 : REM Value of pi
RESOLUTION=2.5                    : REM Size of grid "squares"

REM  BEGIN MAIN PROGRAM
REM * Menu Creation (Program overhead)
MENU 1,0,1,"Control"
MENU 1,1,1,"Stop": MENU 1,2,1,"Start"
MENU 2,0,1,"Grid Points"
MENU 2,1,1,"  100":MENU 2,2,1,"  156"
MENU 2,3,1,"  204":MENU 2,4,1,"  277"
MENU 2,5,1,"  400":MENU 2,6,1,"  625":MENU 2,7,1,"  800"
MENU 2,8,2,"1040":MENU 2,9,1,"1200":MENU 2,10,1,"1600":MENU 2,11,1,"1800"
MENU 2,12,1,"2500":MENU 3,0,1,"Hidden Lines"
MENU 3,1,2,"Solid Sheet":MENU 3,2,1,"Fish Net"
MENU 4,0,1,"Grid Pattern":MENU 4,1,2,"White": MENU 4,2,1,"Black":MENU 4,3,1,"Gra
MENU 5,0,1,"Screen":MENU 5,1,2,"White": MENU 5,2,1,"Black"
LEAVE=0
RESOLUTION=3.1
MENU ON
ON MENU GOSUB MENUCHECK

IDLE: IF LEAVE=1 THEN CONTINUEON
GOTO IDLE

MENUCHECK:
MENUNUMBER=MENU(0):MENUITEM=MENU(1)
IF MENUNUMBER=1 AND MENUITEM=2 THEN BLOCK1 ELSE BLOCK1A
BLOCK1: LEAVE=1: GOTO ENDOFSUB
BLOCK1A: IF MENUNUMBER=1 AND MENUITEM=1 THEN END
IF MENUNUMBER=3 THEN BLOCK2 ELSE BLOCK3
BLOCK2: IF MENUITEM=1 THEN DO1 ELSE DO2
            DO1: MENU 3,1,2 : MENU 3,2,1: MENU 4,0,1:SSFLAG=0: GOTO ENDOFSUB
            DO2: MENU 3,1,1: MENU 3,2,2:  SSFLAG=1:MENU 4,0,0:GOTO ENDOFSUB
BLOCK3: IF MENUNUMBER=2 THEN DO3 ELSE BLOCK4
            DO3:
BLOCK4: IF MENUNUMBER=2 THEN BLOCK5 ELSE BLOCK6
BLOCK5: REM * assign resolution to appropriate number of grid points
             IF MENUITEM=1 THEN RESOLUTION =10
             IF MENUITEM=2 THEN RESOLUTION=8
             IF MENUITEM=3 THEN RESOLUTION=7
             IF MENUITEM=4 THEN RESOLUTION=6
             IF MENUITEM=5 THEN RESOLUTION=5
             IF MENUITEM=6 THEN RESOLUTION=4
             IF MENUITEM=7 THEN RESOLUTION=3.53
             IF MENUITEM=8 THEN RESOLUTION=3.1
             IF MENUITEM=9 THEN RESOLUTION=2.89
             IF MENUITEM=10 THEN RESOLUTION=2.5
             IF MENUITEM=11 THEN RESOLUTION=2.36
             IF MENUITEM=11 THEN RESOLUTION=2
             FOR A=1 TO 12 : MENU 2,A,1 : NEXT A
             MENU 2,MENUITEM,2
BLOCK6: IF MENUNUMBER=4 THEN BB ELSE BLOCK7:
             BB: IF MENUITEM=1 THEN  pat=0 : GOTO EN
                   IF MENUITEM=2 THEN : pat=-1: GOTO EN
                   IF MENUITEM=3 THEN : pat=84!
                   EN: FOR A=1 TO 3 :MENU 4,A,1: NEXT A : MENU 4,MENUITEM,2
BLOCK7: IF MENUNUMBER=5 THEN BC ELSE ENDOFSUB:
             BC: IF MENUITEM=1 THEN MENU 5,1,2: MENU 5,2,1 : SCREA=0
                   IF MENUITEM=2 THEN MENU 5,2,2: MENU 5,1,1 : SCREA=-1
                   FOR A=0 TO 3: PATT%(A)=SCREA: NEXT A: CALL BACKPAT(VARPTR(PA
                   CLS
ENDOFSUB:
RETURN

CONTINUEON:
MENU 1,1,1: MENU 1,2,0
MENU 2,0,0:MENU 3,0,0:MENU 4,0,0:MENU 5,0,0
ON MENU GOSUB MENUCHECK2:
INPUT" Vantage Point (X,Y,Z) ",VX,VY,VZ    : REM Vantage Point
CALL Quadrant(VX,VY,Quad)
CALL Configuration(VX,VY,VZ)                  : REM  Construct Rotation transfor
CALL PLOT                                                 : REM Plot the function
MENUCHECK2:END
STOP
```

continued

```
REM ************************************************************
SUB PLOT STATIC
        SHARED SX0,SY0,Beta,T(),M,RESOLUTION,VR,I,J,PI,T1(),T2(),POLY%(),PATT%()
        CLS:
                                            REM  (NX, NY, NZ) True coordinate syst
                                            REM  (STX, STY) Screen coordinate syst
DIM HOLDX(4),HOLDY(4)      :REM  Arrays to hold four image points' coordinates

X=0 : Y=0 : Z=50
IF pat=-1 THEN COL=30 ELSE COL=33
REM PLOT X,Y,Z AXIS
     IF SCREA=-1 THEN COLOR=30 ELSE COLOR=33
     CALL Transformation(X,Y,Z,STX,STY)
     LINE(SX0,SY0)-(STX,STY),COLOR
     X=50 : Y=0 : Z=0
     CALL Transformation(X,Y,Z,STX,STY)
     LINE(SX0,SY0)-(STX,STY),COLOR
     X=0 : Y=50 : Z=0
     CALL Transformation(X,Y,Z,STX,STY)
     LINE(SX0,SY0)-(STX,STY),COLOR

REM Calculate which quardrant to start plotting ( needed for hidden line removal
NYlosign=1:NYhisign=1:NXlosign=1:NXhisign=1:ressignx=1:ressigny=1

IF Quad=1 THEN 1 ELSE 2
1 : NYlosign=-1 : NXlosign=-1
2 : IF Quad=2 THEN 3 ELSE 4
3 : NYlosign=-1: NXhisign=-1:ressignx=-1
4 : IF Quad=3 THEN 5 ELSE 6
5 : NYhisign=-1 : NXhisign=-1:ressignx=-1:ressigny=-1
6 : IF Quad=4 THEN 7 ELSE 8
7 : NYhisign=-1:NXlosign=-1:ressigny=-1
8 :

FLAG=0  : STX1=121 : STY1=50
FOR NY=NYlosign*50 TO NYhisign*50 STEP RESOLUTION*ressigny
    FOR NX=NXlosign*50 TO NXhisign*50 STEP RESOLUTION*ressignx
REM Construct the plot grid
   GOSUB FUNCTION                                                    : REM Calc
        CALL Transformation(NX,NY,NZ,STX,STY): REM Transform to screen coordinat
        PSET(STX,STY)
        IF NX=NXlosign*50 THEN BELOW ELSE GOON
        BELOW: NY=NY+RESOLUTION*ressigny: REM Calculate first pair of points

    GOSUB FUNCTION: REM Calculate NZ
        CALL Transformation(NX,NY,NZ,STX2,STY2): REM Transform to screen coordin
        PSET(STX2,STY2)
        NY=NY-RESOLUTION*ressigny: REM Reset to current point in for loops
        HOLDX(1)=STX:HOLDX(2)=STX2: REM Remember point
        HOLDY(1)=STY:HOLDY(2)=STY2: REM Remember point
        NX=NX+RESOLUTION*ressignx

        GOON:
     GOSUB FUNCTION                                                    : REM Ca
        CALL Transformation(NX,NY,NZ,STX2,STY2): REM Transform to screen coordin
        PSET (STX2,STY2)
        HOLDX(3)=STX2:HOLDY(3)=STY2: REM Remember point
        NY=NY+RESOLUTION*ressigny

    GOSUB FUNCTION: REM Calculate NZ
        CALL Transformation(NX,NY,NZ,STX2,STY2): REM Transform to screen coordin
        PSET (STX2,STY2)
        HOLDX(4)=STX2:HOLDY(4)=STY2: REM Remember point
        NY=NY-RESOLUTION*ressigny: REM Reset to current point in for loops

        CALL SOLIDSHEET(HOLDY(1),HOLDX(1),HOLDY(2),HOLDX(2),HOLDY(3),HOLDX(3),HO
        HOLDX(1)=HOLDX(3):HOLDY(1)=HOLDY(3): REM Remember first pair
        HOLDX(2)=HOLDX(4):HOLDY(2)=HOLDY(4): REM Remember first pair
        NEXT NX
NEXT NY
GOTO JUMPER

REM Gosub routine to calculate NZ
       FUNCTION:
       NZ=COS( (NX^2+NY^2)/600*PI)*60: REM Insert Function Here <--
       IF NZ<0 THEN NZ=NZ/2
       IF (NX^2+NY^2)>900 THEN NZ=0
       RETURN
JUMPER:
END SUB


REM ************************************************************
SUB Transformation(X1,X2,X3,STTX,STTY) STATIC
```

```
      SHARED T(),VR,SX0,SY0,Beta,I,J
REM N(1)=X point               N(2)=Y point          N(3)=Z point


REM Rotate the coordinate axis with rotation transformation
REM  In other words transform true coordinates to proper coordinates
N(1)=X1:N(2)=X2:N(3)=X3
FOR I=1 TO 3
      NR(I)=0
      FOR J=1 TO 3
            NR(I)=NR(I)+T(I,J)*N(J)
      NEXT J
NEXT I


RDTP=VR-NR(1)          :REM This is the distance to the point if its y,z=0
IF RDTP<0 OR RDTP=0 THEN ohno ELSE Contin
ohno:
REM The point is at the viewer's location or it is behind the viewer
REM  The point can not be plotted
REM An error occurs
STOP

Contin:
REM : Transform proper coordinates into adjusted screen coordinates
TanThetasx=NR(2)/RDTP
TanThetasy=NR(3)/RDTP
SX=SX0*TanThetasx/TAN(Beta)
SY=SX0*TanThetasy/TAN(Beta)

REM: Transform adjusted screen coordinates into screen coordinates
STTX=SX0+SX:STTY=SY0-SY
END SUB



REM*********************************************************
SUB Configuration(VX,VY,VZ)STATIC
        SHARED PI,T(),T1(),T2(),PI,Beta,RS,SX0,SY0,VR,VRpolar,I,J

REM T2 is  the rotation matrix about the z axis
REM T1 is the rotation matrix about the y axis

Beta=PI/4       : REM   Angle the window subtends
RS=10             : REM    Distance viewer is away from window
SX0=250         : REM   1/2 screen size in x direction
SY0=150         : REM    1/2 screen size in y direction

FOR I=1 TO 3:FOR J=1 TO 3:T1(I,J)=0:T2(I,J)=0: NEXT J,I: REM Zero Arrays

REM Calculate needed values
VR=SQR( (VX)^2 + (VY)^2 + (VZ)^2 )
VRpolar=SQR( (VX)^2 + (VY)^2 )

REM Create the two rotation transformation matrices
FOR I=1 TO 3: FOR J=1 TO 3:T1(I,J)=0:T2(I,J)=0: NEXT J,I
T2(1,1)=1:T2(2,2)=1:T2(3,3)=1
IF VRpolar=0 THEN SKIProtation
     T2(1,1)=(VX)/VRpolar
     T2(1,2)=(VY)/VRpolar
     T2(2,1)=-T2(1,2)
     T2(2,2)=T2(1,1)
SKIProtation:
T1(1,1)=VRpolar/VR
T1(1,3)=(VZ)/VR
T1(2,2)=1
T1(3,1)=-T1(1,3)
T1(3,3)=T1(1,1)

REM : Multiply the two matrices together
FOR I=1 TO 3
FOR J=1 TO 3
T(I,J)=0
FOR K=1 TO 3
T(I,J)=T(I,J)+T1(I,K)*T2(K,J)
NEXT K,J,I
END SUB


REM *********************************************************
SUB SOLIDSHEET(A1,B1,A2,B2,A3,B3,A4,B4,COL)STATIC
SHARED POLY%(),PATT%(),SSFLAG,pat,SCREA
```

continued

```
REM This subroutine uses the quick draw machine routines
 CALL PENNORMAL
 CALL SHOWPEN

POLY%(0)=26
POLY%(1)=0 : POLY%(2)=0 : POLY%(3)=500 : POLY%(4)=250
POLY%(5)=A1 : POLY%(6)=B1 : POLY%(7)=A2 : POLY%(8)=B2
POLY%(9)=A4 : POLY%(10)=B4 : POLY%(11)=A3: POLY%(12)=B3
FOR A=0 TO 3 : PATT%(A)=pat : NEXT A
IF SSFLAG=0 THEN CALL FILLPOLY(VARPTR(POLY%(0)),VARPTR(PATT%(0)))
CALL FRAMEPOLY(VARPTR(POLY%(0)))
LINE(B1,A1)-(B3,A3),COL:  LINE(B2,A2)-(B4,A4),COL
 LINE(B1,A1)-(B2,A2),COL:  LINE(B3,A3)-(B4,A4),COL
END SUB


REM*********************************************************
SUB Quadrant(VX,VY,Quad)STATIC
IF VX>0 AND VY>0 THEN Quad=1
IF VX<0 AND VY>0 THEN Quad=2
IF VX<0 AND VY<0 THEN Quad=3
IF VX>0 AND VY<0 THEN Quad=4
END SUB
```

ENTROPY.BAS Accompanies "Information Theory" by Ramachandran Bharath, BYTE, December, 1987, page 291

```
10 REM ENTROPY.BAS FOR IBM PC AND COMPATIBLES
15 REM THIS PROGRAM WAS DEBUFFED AND STREAMLINED BY PROF JUNE PARSONS OF NMU
20 CLS
30 DIM S$(40),PROB(40)
40 LET ENTROPY = 0
50 INPUT "How many different symbols are there in your alphabet ? ",NUMBER
60 PRINT
70 FOR INDEX = 1 TO NUMBER
80 PRINT "Type in symbol #  "; INDEX ; " and relative frequency.  Separate"
90 PRINT " the two using a comma. (REMEMBER , RELATIVE FREQUENCIES "
100 PRINT "MUST ADD UP TO 1.00) "
110 INPUT S$(INDEX),PROB(INDEX)
120 ENTROPY = ENTROPY + PROB(INDEX)*LOG(1/PROB(INDEX) )
130 NEXT INDEX
140 REM ENTROPY CALCULATIONS USE LOGS TO THE BASE 2
150 LET ENTROPY = ENTROPY/LOG(2)
160 PRINT
170  PRINT  "Entropy is  ", ENTROPY
180 PRINT : PRINT
190 PRINT "Symbol","Probability"
200 FOR INDEX = 1 TO NUMBER
210 PRINT S$(INDEX),PROB(INDEX)
220 NEXT INDEX
230 END
```

CARTOG.PAS Accompanies "Mapping the World in Pascal" by Robert Miller and Francis Reddy, BYTE, December 1987, page 329

```
PROGRAM Cartog;
{ This program plots geographic data from the file
  WORLD.DAT and coordinate grids on the Mercator,
  Equidistant Cylindrical, Sinusoidal, Hammer, and
  Orthographic map projections.
}
CONST Sqrt2        = 1.4142135623731;
      PI           = 3.1415926535898;
      HalfPI       = 1.5707963267949;
      TwoPI        = 6.2831853071796;
      Radian       = 1.7453292519943E-2;
      RadianDiv100 = 1.7453292519943E-4; { PI/180/100, needed to convert }
                                         { data in WORLD.DAT to radians  }

CONST XCENTER    : INTEGER = 320;        { CGA Graphics constants.       }
      YCENTER    : INTEGER =  99;        { Screen center X and Y         }
```

```
      ASPECT     : REAL    = 2.4;        ( 640x200 aspect ratio          )
      R          : REAL    =  40;        ( Default map radius            )
      NotVisible : INTEGER = -32767;     ( Flag for point visibility     )

TYPE  LLREC      = RECORD
      CODE       : ARRAY[0..1] OF CHAR;
      LONGI, LATI: INTEGER; END;

VAR   LL         : LLREC;
      LLF        : FILE OF LLREC;

VAR   LastX, LastY, XP, YP    : INTEGER; ( Save variables for plotting )
      COLOR_GLB               : INTEGER;

VAR   I, J, K, MapType, M, X1,Y1,
      X2, Y2, SX, SY, CENTER  : INTEGER;

VAR   L, L1, LONGR, LSTEP,
      B, LATR, BSTEP, X, Y,
      PHI1, Lambda0           : REAL;

VAR   XX, YY, SA, SB          : REAL;

VAR   LastPtVis, GRID         : BOOLEAN;
VAR   CH                      : CHAR;

FUNCTION ArcCos(X: REAL): REAL;
BEGIN
    IF  ABS(X) < 1  THEN ArcCos:= ARCTAN(SQRT(1-SQR(X))/X)
    ELSE  IF X = 1  THEN ArcCos:=  0
    ELSE  IF X =-1  THEN ArcCos:= PI;
END;   ( ArcCos. )

FUNCTION ArcSin(X: REAL): REAL;
BEGIN
    IF ABS(X) < 1  THEN ArcSin:= ARCTAN(X/SQRT(1-SQR(X)))
    ELSE IF X = 1  THEN ArcSin:= HalfPI
    ELSE IF X =-1  THEN ArcSin:=-HalfPI;
END;    ( ArcSin. )

FUNCTION ArcTanH(X : Real):  Real;
VAR A,T :  REAL;
BEGIN
   T:=ABS(X);
   IF T < 1 THEN
   BEGIN
      A := 0.5 * LN((1 + T)/(1 - T));
      IF X < 0 THEN ArcTanH := -A ELSE ArcTanH :=A;
   END;
END;  ( ArcTanH. )

FUNCTION Meridian(Lambda, Lambda0: REAL):REAL;
( Returns difference between current longitude and map center. )
VAR DelLam : REAL;
BEGIN
    DelLam := Lambda - Lambda0;
    IF DelLam < -PI THEN DelLam := DelLam + TwoPI
    ELSE
    IF DelLam >  PI THEN DelLam := DelLam - TwoPI;
    Meridian:=DelLam;
END;   ( Meridian. )

PROCEDURE Mercator(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
( For R = 1: -Pi <= X <= Pi, -Pi/2 <= Y <= Pi/2.  )
CONST MaxLat : REAL = 1.397;   (~80 degrees. )
     ( REAL = 1.483;    ~85 degrees. )
BEGIN
    IF ABS(Phi) < MaxLat THEN
    BEGIN
       Lambda := Meridian(Lambda, Lambda0);
       X := R * Lambda;
       Y := R * ArcTanH(SIN(Phi));
    END
    ELSE X := NotVisible;
END;   ( Mercator. )

PROCEDURE EquiCyl(Lambda, Lambda0, Phi, Phi1, R : REAL; VAR X, Y : REAL);
( For R = 1: -Pi <= X <= Pi, -Pi/2 <= Y <= Pi/2. )
BEGIN
    Lambda := Meridian(Lambda, Lambda0);
    X := R * Lambda * COS(Phi1);
    Y := R * Phi;
END;   ( EquiCyl. )
```

continued

```
PROCEDURE Sinusoidal(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
{ For R = 1:  -Pi <= X <= Pi  and  -Pi/2 <= Y <= Pi/2. }
BEGIN
     Lambda := Meridian(Lambda, Lambda0);
     X := R * Cos(Phi) * Lambda ;
     Y := R * Phi;
END;  { Sinusoidal. }

PROCEDURE Hammer(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
{ For R = 1: -2«2 <= X <=2«2  and  - «2 <= Y <= «2. }
VAR K, CosPhi, HalfLambda        : REAL;
BEGIN
    HalfLambda := 0.5*Meridian(Lambda, Lambda0);
    CosPhi:=COS(Phi);
    K := R * SQRT2 / SQRT(1 +CosPhi * COS(HalfLambda));
    X := 2 * K * CosPhi * (SIN(HalfLambda));
    Y := K * SIN(Phi);
END;  { Hammer. }

PROCEDURE Orthographic(Lambda, Lambda0, Phi, Phi1, R: REAL; VAR X, Y : REAL);
{ For R = 1: -2 <= X,Y <= 2. }
VAR CosC, CosL, SinPhi1, CosPhi1, SinPhi, CosPhi, R2 :  Real;
BEGIN
    Lambda :=Meridian(Lambda, Lambda0);  R2:=R+R;
    CosPhi1:=COS(Phi1);   SinPhi1:=SIN(Phi1);
    CosPhi :=COS(Phi);    SinPhi:= SIN(Phi);
    CosL   :=COS(Lambda)*CosPhi;
    CosC   :=SinPhi1 * SinPhi + CosPhi1 * COSL;
    IF CosC >= 0 THEN
    BEGIN
       X :=R2 * CosPhi * SIN(Lambda);
       Y :=R2 * (CosPhi1 * SinPhi - SinPhi1 * COSL);
    END ELSE X:=NotVisible;
END;  { Orthographic. }

PROCEDURE Beep;
{ Sounds a tone when map is complete. }
BEGIN
    Sound(880);  Delay(250);  NoSound;
END;

PROCEDURE PlotPt(VAR LastPtVis: BOOLEAN);
{ Draws a line from the last point to the current (XP,YP) if it is visible. }
VAR IX,IY: INTEGER;
LABEL XIT;
BEGIN
    IX:=ROUND(XP); IY:=ROUND(YP);
    IF LastPtVis THEN DRAW(LastX,LastY,IX,IY,1);
    LastX:=IX; LastY:=IY;
    LastPtVis:=TRUE;
  XIT:
END; { PlotPt. }

PROCEDURE CoordinateGrid(OUTLINE: BOOLEAN; MapType: INTEGER);
CONST LatitudeSpacing  = 30;
      LongitudeSpacing = 30;

VAR Longitude, Latitude, LatLimit,
    MaxLat, LongIncr, LatIncr      : INTEGER;
VAR LL, PP, A, R2, RA, XN, YN,
    SINDT, COSDT                   : REAL;
BEGIN
    CASE MapType OF
       1: BEGIN   MaxLat:=80;  LongIncr:=360;  LatIncr:=160;    END;
       2: BEGIN   MaxLat:=90;  LongIncr:=360;  LatIncr:=180;    END;
       3: BEGIN   MaxLat:=90;  LongIncr:=360;  LatIncr:=5;      END;
    4..5: BEGIN   Maxlat:=90;  LongIncr:=5;    LatIncr:=5;      END;
    END;  { CASE...}

    LL:=0;  PP:=Phi1;
    IF OUTLINE THEN
    BEGIN
        IF MapType = 5 THEN PP:=0;
        LatLimit:=MaxLat;              { Draw only extreme latitudes }
                                       { to make map outline         }
    END
    ELSE LatLimit:= MaxLat DIV LatitudeSpacing*LatitudeSpacing;

    Latitude:=LatLimit;

    WHILE Latitude >= -LatLimit DO      { Draw parallels }
    BEGIN
       LATR:=Latitude*Radian;
       LastPtVis:=FALSE;
```

```
      Longitude:=-180;
      WHILE Longitude <= 180 DO
      BEGIN
         LONGR:=Longitude*Radian;

         CASE MapType OF
         1: BEGIN MERCATOR(LONGR, LL, LATR, R, X, Y);       END;
         2: BEGIN EQUICYL(LONGR, LL, LATR, PP, R, X, Y);    END;
         3: BEGIN SINUSOIDAL(LONGR, LL, LATR, R, X, Y);     END;
         4: BEGIN HAMMER(LONGR, LL, LATR, R, X, Y);         END;
         5: BEGIN ORTHOGRAPHIC (LONGR, LL, LATR, PP, R, X, Y); END;
         END;  ( CASE...)

         IF X > -300 THEN
         BEGIN
            XP:=ROUND(X*ASPECT)+XCENTER;
            YP:=YCENTER-ROUND(Y);
            PlotPt(LastPtVis);
         END ELSE LastPtVis:=FALSE;

         Longitude:=Longitude+LongIncr;
      END;

      IF OUTLINE THEN
           Latitude:=Latitude-2*MaxLat
      ELSE
           Latitude:=Latitude-LatitudeSpacing;
      END;

      IF OUTLINE THEN LL:=0 ELSE LL:=Lambda0;

      Longitude:=-180;                    ( Draw meridians )

      IF MapType >= 4 THEN MaxLat:=90;
      WHILE Longitude <= 180 DO
      BEGIN
         LONGR:=Longitude*Radian;
         LastPtVis:=FALSE;
         Latitude:=MaxLat;
         WHILE Latitude >= -MaxLat DO
         BEGIN
            LATR:=Latitude*Radian;

            CASE MapType OF
            1: BEGIN MERCATOR(LONGR, LL, LATR, R, X, Y);       END;
            2: BEGIN EQUICYL(LONGR, LL, LATR, PP, R, X, Y);    END;
            3: BEGIN SINUSOIDAL(LONGR, LL, LATR, R, X, Y);     END;
            4: BEGIN HAMMER(LONGR, LL, LATR, R, X, Y);         END;
            5: BEGIN ORTHOGRAPHIC( LONGR, LL, LATR, PP, R, X, Y); END;
            END;  ( CASE...)

         IF X > -300 THEN
         BEGIN
            XP:=ROUND(X*ASPECT)+XCENTER;
            YP:=YCENTER-ROUND(Y);
            PlotPt(LastPtVis);
         END ELSE LastPtVis:=FALSE;

         Latitude:=Latitude-LatIncr;
    END;

    IF OUTLINE THEN
         Longitude:=Longitude+360
    ELSE
         Longitude:=Longitude+LongitudeSpacing;
END;

IF OUTLINE AND (MapType=5) THEN
BEGIN
   A:=0;                              ( Draw circular outline )
   LastPtVis:=False;
   R2:=R + R;
   RA:= R2 * Aspect;
   SINDT:= 0.05996400648;
   COSDT:= 0.99820053993;
   X:=1;   Y:=0;
   XP:= ROUND(XCENTER + RA);
   YP:= ROUND(YCENTER);
   PlotPt(LastPtVis);
   WHILE A <= TwoPI DO
   BEGIN                              ( Compute points on the circle )
      XN:= X * COSDT - Y * SINDT;
      YN:= X * SINDT + Y * COSDT;
```

continued

```
        X:= XN;  Y:= YN;
        XP:= XCENTER + ROUND(X*RA);
        YP:= YCENTER + ROUND(Y*R2);
        PlotPt(LastPtVis);
        A:= A+0.06;
     END; ( While. )
  END;
END;  ( CoordinateGrid. )

PROCEDURE DrawMap(MapType: INTEGER);
VAR Latitude, Longitude : REAL;
VAR LastX : INTEGER;
LABEL XIT;
BEGIN
    LastPtVis:=FALSE; LastX:=0;

    ASSIGN(LLF, 'WORLD.DAT');  RESET(LLF);
    WHILE NOT EOF(LLF) DO
    BEGIN
       READ(LLF, LL);
       IF KeyPressed THEN GOTO XIT;
       LONGR:=LL.LONGI * RadianDiv100;
       LATR :=LL.LATI  * RadianDiv100;

       IF LL.CODE = 'LS' THEN LastPtVis:=FALSE;
       IF (LL.CODE = 'S ') OR (LL.CODE = 'LS') THEN
       BEGIN
          CASE MapType OF
          1: BEGIN MERCATOR(LONGR, Lambda0, LATR, R, X, Y);       END;
          2: BEGIN EQUICYL(LONGR, Lambda0, LATR, Phi1, R, X, Y); END;
          3: BEGIN SINUSOIDAL(LONGR, Lambda0, LATR, R, X, Y);    END;
          4: BEGIN HAMMER(LONGR, Lambda0, LATR, R, X, Y);        END;
          5: BEGIN ORTHOGRAPHIC(LONGR, Lambda0, LATR, Phi1, R, X, Y);  END;
          END;  ( CASE...)

          IF X > -300 THEN
          BEGIN
             XP:=ROUND(X*ASPECT)+XCENTER;
             IF ABS(LastX-XP) > 100 THEN LastPtVis:=FALSE;
             YP:= YCENTER-ROUND(Y);
             PlotPt(LastPtVis);  LastX:=XP;
          END ELSE LastPtVis:=FALSE;
       END;
    END;
 XIT:
END;  ( DrawMap. )

(* --------------------  MAIN  PROGRAM  ------------------ *)

VAR RESP : CHAR;
LABEL XIT;
BEGIN
    MapType:=1;
    WHILE MapType > 0 DO      (*       MENU       *)
    BEGIN
        ClrScr;

        GOTOXY(24,1);  WRITE('C A R T O G');
        LowVideo;
        GOTOXY(1,24);
        WRITE(''':4,'Copyright 1987 by Robert Miller and Francis Reddy');
        GOTOXY(1,3);
        WRITELN(' ':4,'To PLOT:  Choose a projection.  Enter the Central ');
        WRITELN(' ':4,'Meridian of the map (180 to -180 degrees, longitudes');
        WRITELN(' ':4,'west of Greenwich negative). If applicable, enter');
        WRITELN(' ':4,'the Standard Parallel (90 to -90 degrees, southern');
        WRITELN(' ':4,'latitudes negative). The file WORLD.DAT must also be');
        WRITELN(' ':4,'on the logged drive. A tone means the map is done.');
        WRITELN;
        WRITELN(' ':4,'Any key ABORTS plot.  Hit return to restore MENU.');
        NormVideo;
        WRITELN;
        WRITELN;
        WRITE(' ':6,'1. Mercator');
        WRITELN(' ':21,'4. Hammer');
        WRITE(' ':6,'2. Equidistant Cylindrical');
        WRITELN(' ':6,'5. Orthographic');
        WRITELN(' ':6,'3. Sinusoidal');
        WRITELN;
        WRITE(' ':8,'Projection number (1-5) or 0 to quit:    ');
        READLN(MapType);
        If MapType = 0 THEN GOTO XIT;
```

```
        WRITELN;
        WRITE(' ':8,'Central Longitude of Map (default = 0): ');
        Lambda0:=0;
        READLN(Lambda0);  Lambda0:=Lambda0*Radian;

        IF (MapType = 2) OR (MapType = 5) THEN
        BEGIN
            WRITE(' ':8,'Central Latitude  of Map (default = 0): ');
            Phi1:=0;  READLN(Phi1);
            IF Phi1 = 90 THEN Phi1 := HalfPI
            ELSE
            Phi1:=Phi1*Radian;
        END;

        IF MapType >= 4 THEN R:=83 ELSE R:=70;
        WRITE(' ':8,'Plot grid, continents or both (G/C/B)?  ');
        READLN(RESP);  RESP:=UPCASE(RESP);
        GRID:=(RESP ='G') OR (RESP = 'B');

        HiRes;  HiResColor(15);              { Set CGA Graphics Mode }

        IF GRID THEN CoordinateGrid(FALSE,  MapType);
        CoordinateGrid(TRUE, MapType);

        IF (RESP = 'B') OR (RESP = 'C') THEN DrawMap(MapType);
      Beep;
    XIT:

    IF MapType > 0 THEN
       While NOT KeyPressed DO ;             { Wait for key strike }

    TEXTMODE(BW80);                          { Return to Text Mode }
    ClrScr;
    END;  { WHILE MapType > 0...}
END.
```

CPLOT.PAS Accompanies "Mapping the World in Pascal" by Robert Miller and Francis Reddy, BYTE, December 1987, page 329

```
PROGRAM MapProjections;
{  Map projection program for Hewlett-Packard 7475 plotter. }
{$V-}
CONST Sqrt2    = 1.4142135623710;
      PI       = 3.141592653589793238;
      HalfPI   = 1.570796326794897;
      ThreePI2= 4.71238898038469;
      TwoPI    = 6.283185307179587;
      Degree   = 57.29577951308232088;
      Radian   = 0.01745329251994329577;

CONST XCENTER : REAL    = 6.2;       { CGA Graphics constants. }
      YCENTER : REAL    = 5.1;
      ASPECT  : REAL    =   1;
      R       : REAL    =   1;

TYPE  S80       = STRING[80];
TYPE  LLREC     = RECORD
      CODE        : ARRAY[0..1] OF CHAR;
      LONGI, LATI : INTEGER; END;

VAR   LL          : LLREC;
      LLF         : FILE OF LLREC;
      PLT         : TEXT[$1000];
VAR   FNAME       : STRING[64];

VAR   LastX, LastY, XP, YP  : REAL;  { Save variables for plotting. }
      COLOR_GLB, IPEN       : INTEGER;
      SIZE                  : CHAR;

VAR   I, J, K, MapType, M, X1,Y1, X2, Y2,
      SX, SY, CENTER        : INTEGER;

VAR   L, L1, LONGR, LSTEP,
      B, LATR, BSTEP, X, Y,
      PHI1, Lambda0         : REAL;

VAR XX, YY, SA, SB          : REAL;
```

continued

```
VAR   LastPtVis, GRID        : BOOLEAN;
VAR   CH                     : CHAR;

PROCEDURE Initialize;
BEGIN
    WRITE(' ':13, 'Enter map size (A/B): '); READLN(SIZE);
    SIZE:=UPCASE(SIZE);;

    ASSIGN(PLT, 'WORLD.PLT');  REWRITE(PLT);
    WRITELN(PLT, 'IN;');
    IF SIZE = 'A' THEN WRITELN(PLT,'PS4; IP 100,100,868,868;')
    ELSE  WRITELN(PLT, 'PS0; IP 100,100,1124,1124;');
    WRITELN(PLT, 'SC 0,1,0,1;');
    WRITELN(PLT, 'VS 4; SP 3;');
END;  ( Initialize. )

PROCEDURE LabelMap(MapType : INTEGER);
VAR TITLE  : STRING[80];
VAR XT, YT : REAL;
BEGIN
     CASE MapType OF
     1: BEGIN TITLE:='Mercator';                    END;
     2: BEGIN TITLE:='Equidistant Cylindrical';     END;
     3: BEGIN TITLE:='Miller';                      END;
     4: BEGIN TITLE:='Sinusoidal';                  END;
     5: BEGIN TITLE:='Hammer';                      END;
     6: BEGIN TITLE:='Orthographic' ;               END;
     7: BEGIN TITLE:='Stereographic';               END;
     END;

     TITLE:=TITLE + ' Map Projection';
     IF SIZE = 'A' THEN XT:=12.38
     ELSE XT:= 14;
     YT:= YCENTER-4.5;
     WRITELN(PLT, 'DI 0,1; SR 18,18; PU',XT:7:2, ',', YT:7:2);
     WRITELN(PLT, 'LB' + TITLE + CHR(3)+';');

     XT:=XT +0.24;
     WRITELN(PLT, 'PU ',XT:7:2, ',', YT:7:2, '; SR 14,14;');
     WRITELN(PLT, 'LBCentral Meridian ',ROUND(Lambda0*Degree):4,
                  ' degrees'+CHR(3)+';');

     IF (MapType = 2) OR (MapType = 6) OR (MapType = 7) THEN
     BEGIN
         XT:=XT +0.24;
         WRITELN(PLT, 'PU ',XT:7:2, ',', YT:7:2, ';');
         WRITELN(PLT, 'LBCentral Latitude ',ROUND(Phi1*Degree):4,
                      ' degrees'+CHR(3)+';');
     END;

     XT:=XT +0.24;
     WRITELN(PLT, 'SR 10,10; PU', XT:7:2 ,',', YT:7:2,';');
     WRITELN(PLT, 'LBRobert D. Miller, 1986.' +CHR(3)+';');
END;  ( LabelMap. )

FUNCTION ArcCos(X: REAL): REAL;
BEGIN
    IF  ABS(X) < 1  THEN ArcCos:= ARCTAN(SQRT(1-SQR(X))/X)
    ELSE  IF X = 1  THEN ArcCos:=  0
    ELSE  IF X =-1  THEN ArcCos:= PI;
END;   ( ArcCos. )

FUNCTION ArcSin(X: REAL): REAL;
BEGIN
    IF ABS(X) < 1  THEN ArcSin:= ARCTAN(X/SQRT(1-SQR(X)))
    ELSE IF X = 1  THEN ArcSin:= HalfPI
    ELSE IF X =-1  THEN ArcSin:=-HalfPI;
END;    ( ArcSin. )

FUNCTION ArcTanH(TERM : Real):  Real;   (* Inverse hyperbolic tangent *)
VAR A,T :  real;
BEGIN
    T:=ABS(TERM);
    IF T < 1 THEN
    BEGIN
        A := 0.5 * LN((1 +T)/(1 -T));
        IF TERM < 0 THEN ArcTanH := -A ELSE ArcTanH :=A;
    END;
END;  ( ArcTanH. )

(*
        Map Projection Library
        by Robert Miller and Francis Reddy, 20 October 1986.
        The following routines are based on equations found
```

```
        in the U.S. Geological Survey's Bulletin 1532,
        "Map Projections Used by the U.S. Geological Survey"
        by John P. Snyder and "Introduction to Map Projections"
        by Porter McDonnell, Jr.
*)
FUNCTION Meridian(Lambda, Lambda0: REAL):REAL;
{ Returns difference between current longitude and map center. }
VAR DelLam : REAL;
BEGIN
    DelLam := Lambda - Lambda0;
    IF DelLam < -PI THEN DelLam := DelLam + TwoPi
    ELSE
    IF DelLam >  PI THEN DelLam := DelLam - TwoPi;
    Meridian:=DelLam;
END;   { Meridian. }

PROCEDURE Mercator(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
{  For R = 1: -Pi <= X <= Pi, -Pi/2 <= Y <= Pi/2.  }
CONST MaxLat : REAL = 1.397;   {~80 degrees. }
             { REAL = 1.483;    ~85 degrees. }
BEGIN
    IF ABS(Phi) < MaxLat THEN
    BEGIN
        Lambda := Meridian(Lambda, Lambda0);
        X := R * Lambda;
        Y := R * ArcTanH(SIN(Phi));
    END
    ELSE X := -32767;
END;   { Mercator. }

PROCEDURE EquiCyl(Lambda, Lambda0, Phi, Phi1, R : REAL; VAR X, Y : REAL);
{  For R = 1: -Pi <= X <= Pi, -Pi/2 <= Y <= Pi/2. }
BEGIN
    Lambda := Meridian(Lambda, Lambda0);
    X := R * Lambda * COS(Phi1);
    Y := R * Phi;
END;   { EquiCyl. }

PROCEDURE Miller(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
{ For R = 1: -Pi <= X <= Pi, -Pi/2 <= Y <= Pi/2. }
BEGIN
     Lambda := Meridian(Lambda, Lambda0);
     X := R * Lambda;
     Y := R * ArcTanH(SIN(0.8 * Phi)) * 1.25;
END;   { Miller. }

PROCEDURE Sinusoidal(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
{ For R = 1:  -Pi <= X <= Pi  and  -Pi/2 <= Y <= Pi/2. }
BEGIN
     Lambda := Meridian(Lambda, Lambda0);
     X := R * Cos(Phi) * Lambda ;
     Y := R * Phi;
END;  { Sinusoidal. }

PROCEDURE Hammer(Lambda, Lambda0, Phi, R : REAL; VAR X, Y : REAL);
{  For R = 1: -2«2 <= X <=2«2  and  -«2 <= Y <= «2. }
VAR K, CosPhi, HalfLambda        : REAL;
BEGIN
    HalfLambda := 0.5*Meridian(Lambda, Lambda0);
    CosPhi:=COS(Phi);
    K := R * SQRT2 / SQRT(1 +CosPhi * COS(HalfLambda));
    X := 2 * K * CosPhi * (SIN(HalfLambda));
    Y := K * SIN(Phi);
END;  { Hammer. }

PROCEDURE Orthographic(Lambda, Lambda0, Phi, Phi1, R: REAL; VAR X, Y : REAL);
{  For unit radius R of generating globe, R = 1,  -2 <= X,Y <= 2. }
VAR CosC, CosL, SinPhi1, CosPhi1, SinPhi, CosPhi, R2 :  Real;
BEGIN
    Lambda :=Meridian(Lambda, Lambda0);  R2:=R+R;
    CosPhi1:=COS(Phi1);   SinPhi1:=SIN(Phi1);
    CosPhi :=COS(Phi);    SinPhi:= SIN(Phi);
    CosL   :=COS(Lambda)*CosPhi;
    CosC   :=SinPhi1 * SinPhi + CosPhi1 * COSL;
    IF CosC >= 0 THEN
    BEGIN
        X :=R2 * CosPhi * SIN(Lambda);
        Y :=R2 * (CosPhi1 * SinPhi - SinPhi1 * COSL);
    END ELSE X:=-32767;
END;  { Orthographic. }
```

continued

```
PROCEDURE Stereographic(Lambda, Lambda0, Phi, Phi1, R : REAL; VAR X, Y :REAL);
{  For R = 1, -2 <= X,Y <= 2. }
VAR K, CosC, CosL, CosPhi, SinPhi, SinPhi1, CosPhi1  : Real;
BEGIN
    Lambda :=Meridian(Lambda, Lambda0);
    IF (Lambda <> Lambda0 + TwoPi) OR (Lambda <> Lambda0-TwoPi) THEN
    BEGIN
        SinPhi:= SIN(Phi);     CosPhi:= COS(Phi);
        SinPhi1:=SIN(Phi1);    CosPhi1:=COS(Phi1);
        CosL:=COS(Lambda) * CosPhi;
        CosC :=SinPhi1 * SinPhi + CosPhi1 * COSL;
        IF CosC >= 0 THEN
        BEGIN
            K := R * 2 / (1 + CosC);
            X := K * CosPhi * SIN(Lambda);
            Y := K * (CosPhi1 * SinPhi - SinPhi1 * COSL) ;
        END
        ELSE X:=-32767;
    END;
END;  { Stereographic. }

PROCEDURE PlotPt(VAR LastPtVis: BOOLEAN);
{   Draws a line from the last point to the current (XP,YP) if it is visible. }
LABEL XIT;
BEGIN
    IF LastPtVis THEN  WRITELN(PLT, 'PD',XP:7:3, ',', YP:7:3,';')
    ELSE WRITELN(PLT, 'PU',XP:7:3, ',', YP:7:3,';');
    LastX:=XP; LastY:=YP; LastPtVis:=TRUE;
  XIT:
END; { PlotPt. }

PROCEDURE CoordinateGrid(OUTLINE: BOOLEAN; MapType: INTEGER);
VAR Longitude, Latitude, MaxLat, LongIncr, LatIncr : INTEGER;
VAR LL, PP, A, RA, R2    : REAL;
VAR X, Y, XN, YN, SINDT, COSDT  : REAL;
BEGIN
    CASE MapType OF
       1: BEGIN   MaxLat:=75;  LongIncr:=360;  LatIncr:=160;    END;
       2: BEGIN   MaxLat:=90;  LongIncr:=360;  LatIncr:=180;    END;
       3: BEGIN   MaxLat:=90;  LongIncr:=360;  LatIncr:=5;      END;
    4..5: BEGIN   MaxLat:=75;  LongIncr:=5;    LatIncr:=5;      END;
    END;  { CASE...}

    LL:=0;  PP:=Phi1;
    IF OUTLINE THEN
    BEGIN IF MapType >  1 THEN MaxLat:=90
          ELSE MaxLat:=80;
          IF MapType >= 5 THEN PP:=0;
    END;

    Latitude:=MaxLat;

    WHILE Latitude >= -MaxLat DO       { Draw parallels }
    BEGIN
        LATR:=Latitude*Radian;
        LastPtVis:=FALSE;

        Longitude:=-180;
        WHILE Longitude <= 180 DO
        BEGIN
            LONGR:=Longitude*Radian;

            CASE MapType OF
            1: BEGIN MERCATOR(LONGR, LL, LATR, R, X, Y);       END;
            2: BEGIN EQUICYL(LONGR, LL, LATR, PP, R, X, Y);    END;
            3: BEGIN MILLER(LONGR, LL, LATR, R, X, Y);         END;
            4: BEGIN SINUSOIDAL(LONGR, LL, LATR, R, X, Y);     END;
            5: BEGIN HAMMER(LONGR, LL, LATR, R, X, Y);         END;
            6: BEGIN ORTHOGRAPHIC( LONGR, LL, LATR, PP, R, X, Y); END;
            7: BEGIN STEREOGRAPHIC(LONGR, LL, LATR, PP, R, X, Y); END;
            END;  { CASE...}

            IF X > -300 THEN
            BEGIN
                XP:=X+XCENTER;
                YP:=Y+YCENTER;
                PlotPt(LastPtVis);
            END ELSE LastPtVis:=FALSE;

            Longitude:=Longitude+LongIncr;
        END;

        IF OUTLINE THEN Latitude:=Latitude-2*MaxLat
        ELSE Latitude:=Latitude-15;     { Latitude:=Latitude-10; }
    END;
```

```
    IF OUTLINE THEN LL:=0 ELSE LL:=Lambda0;

    Longitude:=-180;                    ( Draw meridians )
    IF OUTLINE AND (MapType > 5) THEN Longitude:=-90;
    IF MapType >= 4 THEN MaxLat:=90;
    WHILE Longitude <= 180 DO
    BEGIN
        LONGR:=Longitude*Radian;
        LastPtVis:=FALSE;
        Latitude:=MaxLat;
        WHILE Latitude >= -MaxLat DO
        BEGIN
            LATR:=Latitude*Radian;

            CASE MapType OF
            1: BEGIN MERCATOR(LONGR, LL, LATR, R, X, Y);       END;
            2: BEGIN EQUICYL(LONGR, LL, LATR, PP, R, X, Y);    END;
            3: BEGIN MILLER(LONGR, LL, LATR, R, X, Y);         END;
            4: BEGIN SINUSOIDAL(LONGR, LL, LATR, R, X, Y);     END;
            5: BEGIN HAMMER(LONGR, LL, LATR, R, X, Y);         END;
            6: BEGIN ORTHOGRAPHIC( LONGR, LL, LATR, PP, R, X, Y); END;
            7: BEGIN STEREOGRAPHIC(LONGR, LL, LATR, PP, R, X, Y); END;
            END;  ( CASE...)

            IF X > -300 THEN
            BEGIN
                XP:=X+XCENTER;
                YP:=Y+YCENTER;
                PlotPt(LastPtVis);
            END ELSE LastPtVis:=FALSE;

            Latitude:=Latitude-LatIncr;
        END;

        IF OUTLINE THEN
            IF MapType <= 5 THEN Longitude:=Longitude+360
            ELSE
            Longitude:=Longitude+180
        ELSE Longitude:=Longitude+15;
    END;

    IF NOT OUTLINE AND (MapType = 6) THEN ( Draw circular outline. )
    BEGIN
        A:=0;     LastPtVis:=FALSE;
        R2:=R+R;  RA:=R2*Aspect;
        SINDT:=0.05996400648;
        COSDT:=0.99820053993;

        X:=1;  Y:=0;
        XP:=ROUND(XCENTER+RA);  YP:=ROUND(YCENTER);
        PlotPt(LastPtVis);

        WHILE A <= 6.28318 DO
        BEGIN
            XN:= X*COSDT - Y*SINDT;
            YN:= X*SINDT + Y*COSDT;
            X:=XN;  Y:=YN;
            XP:=XCENTER + ROUND(X*RA);  YP:=YCENTER + ROUND(Y*R2);
            PlotPt(LastPtVis);
            A:=A+0.06;
        END;
    END;
END;  ( CoordinateGrid. )

PROCEDURE DrawMap(MapType: INTEGER);
VAR Latitude, Longitude : REAL;
VAR LastX : REAL;
BEGIN
    LastPtVis:=FALSE;  LastX:=0;  LastY:=0;

    IF Fname = '' THEN
    ASSIGN(LLF, 'EARTH.LAT')
    ELSE ASSIGN(LLF, Fname);

    RESET(LLF);

    WHILE NOT EOF(LLF) DO
    BEGIN
        READ(LLF, LL);
        LONGR:=LL.LONGI* 1.74532925199432957E-4;
        LATR :=LL.LATI * 1.74532925199432957E-4;
```

continued

```
        IF LL.CODE = 'LS' THEN LastPtVis:=FALSE;
        IF (LL.CODE = 'S ') OR (LL.CODE = 'LS') THEN
        BEGIN
            CASE MapType OF
            1: BEGIN MERCATOR(LONGR, Lambda0, LATR, R, X, Y);       END;
            2: BEGIN EQUICYL(LONGR, Lambda0, LATR, Phi1, R, X, Y); END;
            3: BEGIN MILLER(LONGR, Lambda0, LATR, R, X, Y);         END;
            4: BEGIN SINUSOIDAL(LONGR, Lambda0, LATR, R, X, Y);     END;
            5: BEGIN HAMMER(LONGR, Lambda0, LATR, R, X, Y);         END;
            6: BEGIN ORTHOGRAPHIC(LONGR, Lambda0, LATR, Phi1, R, X, Y);  END;
            7: BEGIN STEREOGRAPHIC(LONGR, Lambda0, LATR, Phi1, R, X, Y); END;
            END;  ( CASE...)

             IF X > -300 THEN
             BEGIN
                 XP:=X+XCENTER;
                 IF ABS(LastX-XP) > 0.4 THEN LastPtVis:=FALSE;
                 YP:=Y+YCENTER;
                 PlotPt(LastPtVis);  LastX:=XP;
              END ELSE LastPtVis:=FALSE;
        END;
    END;
END;  ( DrawMap. )

(* ---------------------  MAIN  PROGRAM  ----------------- *)
VAR RESP : CHAR;
LABEL XIT;
BEGIN
    LastPtVis:=FALSE;  LastX:=0;  LastY:=0;  IPEN:=-1;
    MapType:=1;
    Fname:='';
    Fname:='WORLD4.DAT';

(*                              MENU                                  *)
    WHILE MapType > 0 DO
    BEGIN
        ClrScr;

        GOTOXY(30,1);  WRITE('GLOBE MAP PROJECTIONS');
        GOTOXY(32,3);  WRITELN('SELECT PARAMETERS');
        WRITELN;
        WRITELN(' ':10,'1. Mercator');
        WRITELN(' ':10,'2. Equidistant Cylindrical');
        WRITELN(' ':10,'3. Miller Cylindrical');
        WRITELN(' ':10,'4. Sinusoidal');
        WRITELN(' ':10,'5. Hammer-Aitoff');
        WRITELN(' ':10,'6. Orthographic');
        WRITELN(' ':10,'7. Stereographic');

        WRITELN;
        WRITE(' ':13,'Projection number (1-7) or 0 to quit: ');
        READLN(MapType);
        If MapType = 0 THEN GOTO XIT;

        CASE MapType OF
        1: BEGIN R:=1.82;   END;
        2: BEGIN R:=1.82;   END;
        3: BEGIN R:=1.82;   END;
        4: BEGIN R:=1.82;   END;
        5: BEGIN R:=2.04;   END;
        6: BEGIN R:=2.18;   END;
        7: BEGIN R:=2.18;   END;
        END;

        WRITELN;
        WRITE(' ':13,'Central Longitude of Map (degrees, default = 0): ');
        Lambda0:=0;
        READLN(Lambda0);  Lambda0:=Lambda0*Radian;

        IF (MapType = 2) OR (MapType = 6) OR (MapType = 7) THEN
        BEGIN
            WRITE(' ':13,'Central Latitude  of Map (degrees, -90 - 90): ');
            Phi1:=0;  READLN(Phi1);  Phi1:=Phi1*Radian;
        END;

        WRITE(' ':13, 'Plot grid, continents or both (G/C/B)? ');
        READLN(RESP); RESP:=UPCASE(RESP);
        GRID:=(RESP ='G') OR (RESP = 'B');

        Initialize;

        IF GRID THEN
        BEGIN
            WRITELN(PLT,'SP 2;');
            CoordinateGrid(FALSE,  MapType);
        END;
```

```
        WRITELN(PLT,'SP 3;');
        CoordinateGrid(TRUE, MapType);

        WRITELN(PLT, 'SP 1');
        IF (RESP = 'B') OR (RESP = 'C') THEN DrawMap(MapType);

      (  LabelMap(MapType);  )

      WRITELN(PLT, 'PU 0,0; SP 0;');
      CLOSE(PLT);
  XIT:
    END;  ( WHILE MapType > 0...)
END.
```

READ.ME Accompanies "Mapping the World in Pascal" by Robert Miller and Francis Reddy, BYTE, December 1987, page 329

To use Cartog.pas with an EGA or Hercules board, you will need to use the follow

EGA Constants:

```
CONST XCENTER = 320;
      YCENTER = 174;
      ASPECT  = 1.37;
      R       = 70;
```

Hercules Constants:

```
CONST XCENTER = 360;
      YCENTER = 174;
      ASPECT  = 1.5;
      R       = 70;
```

Also you will need a software driver for these boards. For example, if you want The following three lines go at the top of the program, right after the line "Program Cartog;"

```
{$I Typedef.sys)
{$I graphix.sys)
{$I Kernel.sys)
```

replace "HiRes; HiResColor(15);" with the following four lines:

```
Initgraphic;
defineWorld(1,0,348,719,0);
selectWorld(1);
selectWindow(1);
```

Finally replace TEXTMODE(BW80); with Leavegraphic;

Setup for the plotter:

Our plots were made with a Hewlett-Packard 7475. A separate version of CARTOG was made to output to the plotter. The constants that we used were:

```
CONST XCENTER = 6.2; ( Page center for B size plot )
      YCENTER = 5.1;
      ASPECT  = 1;  ( Scale factors are the same in both axes    )
      R       = 1;  ( Unit radius, scale in the plotting routine )
```

We also added a plotter initialization procedure:

```
PROCEDURE Initialize;
( PLT is a text file containing the plotter commands.
  At the termination of this program, the file is copied to the
  plotter
)
BEGIN
   WRITE(' ':13, 'Enter map size (A/B): '); READLN(SIZE);
   SIZE:=UPCASE(SIZE);

   ASSIGN(PLT, 'WORLD.PLT');  REWRITE(PLT);

   WRITELN(PLT, 'IN;');  ( Command to initialize plotter )
                         ( Set paper size and scaling points, so
                           plotter effectively works in inches
                           for B size plots and a little less for
                           A size plots )
   IF SIZE = 'A' THEN
        WRITELN(PLT,'PS4; IP 100,100,868,868;')
```

continued

```
    ELSE
        WRITELN(PLT, 'PS0; IP 100,100,1124,1124;');
    WRITELN(PLT, 'SC 0,1,0,1;');
                        { Set pen speed (slow) and select pen 3 }
    WRITELN(PLT, 'VS 4; SP 3;');

END;                    { Initialize. }
```

Finally, we substituted the following for PROCEDURE PlotPt:

```
PROCEDURE PlotPt(VAR LastPtVis: BOOLEAN);
{  Draws a line from the last point to the current (XP,YP) if it
   is visible. }
BEGIN
    IF LastPtVis THEN           { Pen down, go to XP, YP }
        WRITELN(PLT, 'PD', XP:7:3, ',', YP:7:3,';')
    ELSE                        { Pen up,   go to XP, YP }
        WRITELN(PLT, 'PU' , XP:7:3, ',', YP:7:3,';');

    LastX:=; LastY:=YP;
          LastPtVis:=TRUE;
END; { PlotPt. }
```

3-D.PAS Accompanies "Mimicking Mountains," by Tom Jeffery,
BYTE, December 1987, page 337

```
program THREEDEE;
(Wireframe or shaded representation of a fractal surface)
 uses
  quickdraw1, quickdraw2;
 const
  size = 64; (Max array index)
 type
  surface = array[0..size, 0..size] of longint;

 var
  srf : surface;
  ans : string[10];
  srfile : file of longint;
  col, row, range : longint;
  Pt, pt2 : point;
  rct : rect;
  solar : array[1..3] of real;
  az, alt : real;
  Grayscale : array[0..8] of pattern;

 procedure SetUpDrawing;
  var
   R : Rect;
 begin
  HideAll;
  SetRect(R, 0, 38, 511, 341);
  SetDrawingRect(R);
  ShowDrawing
 end;

 function rangefind : longint;
(Finds the difference between the highest)
(and lowest points on the surface)
  var
   min, max, r, c : longint;
 begin
  min := maxlongint;
  max := -maxlongint;
  for r := 0 to size do
   for c := 0 to size do
    begin
     if srf[r, c] < min then
      min := srf[r, c];
     if srf[r, c] > max then
      max := srf[r, c];
    end;
  rangefind := max - min;
 end;

 function proj (r, c : longint) : point;
(Converts row/col/height coord into point on screen)
  var
   z : longint;
   pt : point;
```

```
 begin
  z := (100 * srf[r, c]) div range; {Height}
  pt.h := (4 * c) + 3 * r + 10; {Horiz screen coord}
  pt.v := 70 - (z - 2 * r); {Vert screen coord}
  proj := pt;
 end;

 procedure shade (row, col : longint);
{Selects a gray shade for a patch}
  var
   i : longint;
   dim, ill, normlen : real;
   normal : array[1..3] of real;
 begin
  dim := 100 / range;
{Cross product of two vectors}
  normal[1] := -dim * (srf[row, col] - srf[row + 1, col]);
  normal[2] := -dim * (srf[row, col] - srf[row, col + 1]);
  normal[3] := 1;
  normlen := sqrt(sqr(normal[1]) + sqr(normal[2]) + sqr(normal[3]));{Vector
length}
  for i := 1 to 3 do
   normal[i] := normal[i] / normlen; {Normalize vector}
  ill := 0;
  for i := 1 to 3 do
   ill := ill + solar[i] * normal[i]; {Dot product of normal and solar}
  if ill < 0 then
   penpat(grayscale[0])
  else
   penpat(grayscale[round(ill * 7.9)]); {Set gray level}
 end;

 procedure shadeframe;
{Shades surface}
  var
   r, c : longint;
   start, curr : point;
   patch : polyhandle;
 begin
  for r := 0 to size - 1 do
   for c := 0 to size - 1 do
    begin
     patch := openpoly; {Open polygon for patch}
{Define patch}
     start := proj(r, c);
     moveto(start.h, start.v);
     curr := proj(r, c + 1);
     lineto(curr.h, curr.v);
     curr := proj(r + 1, c + 1);
     lineto(curr.h, curr.v);
     curr := proj(r + 1, c);
     lineto(curr.h, curr.v);
     lineto(start.h, start.v);
     closepoly;
     shade(r, c); {Get shade of patch}
     paintpoly(patch); {Color patch}
     killpoly(patch);
    end;
 end;

 procedure wireframe;
{Draws wireframe of surface}
  var
   r, c : longint;
   start, curr : point;
   patch : polyhandle;
 begin
  setupdrawing;
  for r := 0 to size - 1 do
   for c := 0 to size - 1 do
    begin
     patch := openpoly; {Open polygon for patch}
{Define patch}
     start := proj(r, c);
     moveto(start.h, start.v);
     curr := proj(r, c + 1);
     lineto(curr.h, curr.v);
     curr := proj(r + 1, c + 1);
     lineto(curr.h, curr.v);
     curr := proj(r + 1, c);
     lineto(curr.h, curr.v);
     lineto(start.h, start.v);
     closepoly;
```

continued

```
      penpat(white);
      paintpoly(patch); {Cover up patches behind}
      penpat(black);
      framepoly(patch); {Outline patch}
      killpoly(patch);
     end;
   savedrawing('ramdisk:surf'); {Save drawing to disk}
  end;

  procedure shaded;
 {Gets solar vector, and shades surface}
  begin
   write('Solar altitude?');
   readln(alt);
   alt := alt * 3.14159 / 180;
   write('Solar azimuth?');
   readln(az);
   az := az * 3.14159 / 180;
 {Convert az/alt to three component unit vector}
   solar[3] := sin(alt);
   solar[2] := sin(az) * cos(alt);
   solar[1] := cos(az) * cos(alt);
   setupdrawing;
   shadeframe; {Shade surface}
   savedrawing('surf'); {Save drawing to disk}
  end;

 begin
 {Define array of patterns in order of darkness}
  grayscale[8] := white;
  grayscale[0] := black;
  stuffhex(@grayscale[7], '8800220088002200');
  stuffhex(@grayscale[6], 'AA00AA00AA00AA00');
  stuffhex(@grayscale[5], 'AA11AA44AA11AA44');
  stuffhex(@grayscale[4], 'AA55AA55AA55AA55');
  stuffhex(@grayscale[3], 'BB55EE55BB55EE55');
  stuffhex(@grayscale[2], 'FF55FF55FF55FF55');
  stuffhex(@grayscale[1], 'FF77FFDDFF77FFDD');

  open(srfile, oldfileName('Surface File'));
  for row := 0 to size do
   for col := 0 to size do
    read(srfile, srf[row, col]);
  close(srfile);
  range := rangefind;
  repeat
   showtext;
   write('(w)ire, (s)haded, or (q)uit?');
   readln(ans);
   if ans = 'w' then
    wireframe
   else if ans = 's' then
    shaded;
  until ans = 'q';
 end.
```

TEXTURE.PAS Accompanies "Mimicking Mountains," by Tom Jeffery,
BYTE, December 1987, page 337

```
program textures;
 const
  size = 64;

 var
  srf : array[0..size, 0..size] of longint;
  ans : string[30];
  srfile : file of longint;
  col, row : longint;

 procedure SetUpDrawing;
  var
   R : Rect;
 begin
  HideAll;
  SetRect(R, 0, 38, 511, 341);
  SetDrawingRect(R);
  ShowDrawing
 end;
```

```
 procedure paintpt (row, col : longint);
  var
   pt, pt2 : point;
   rct : rect;
 begin
  pt.v := row;
  pt.h := col;
  pt2.v := row + 1;
  pt2.h := col + 1;
  rct.topleft := pt;
  rct.botright := pt2;
  paintrect(rct);
 end;

 procedure CoastDisp;
{Displays coastline}
  var
   col, row, int : longint;
 begin
  for row := 0 to size do
   for col := 0 to size do
    begin
     if srf[row, col] > 0 then
      penpat(black)
     else
      penpat(white);
     paintpt(row, col);
    end;
 end;

 procedure StripeDisp;
{Displays stripes for heights defined by int}
  var
   col, row, int : longint;
 begin
  write('Interval?');
  readln(int);
  if int > 0 then
   begin
    for row := 0 to size do
     for col := 0 to size do
      begin
       if odd(srf[row, col] div int) then
        penpat(black)
       else
        penpat(white);
       paintpt(row, col);
      end;
   end;
 end;

 procedure TopoDisp;
{Displays contours separated by int}
  var
   col, row, int : longint;
 begin
  write('Interval?');
  readln(int);
  if int > 0 then
   begin
    for row := 0 to size - 1 do
     for col := 0 to size - 1 do
      begin
       if ((srf[row, col] div int) = (srf[row, col + 1] div int)) and
((srf[row, col] div int) = (srf[row + 1, col] div int)) then
        penpat(white)
       else
        penpat(black);
       paintpt(row, col);
      end;
   end;
 end;

 procedure GrainDisp;
{Displays wood grain, grain spacing: int}
  var
   col, row, int, res, sp, drk, gr, lt : longint;
 begin
  write('Interval?');
  readln(int);
  if int > 0 then
   begin
    sp := int div 7;
    drk := sp * 2;
```

continued

```
    gr := sp * 3;
    lt := sp * 4;
    for row := 0 to size do
     for col := 0 to size do
      begin
       res := srf[row, col] mod int;
       if res < sp then
        penpat(black)
       else if res < drk then
        penpat(dkgray)
       else if res < gr then
        penpat(gray)
       else if res < lt then
        penpat(ltgray)
       else
        penpat(white);
       paintpt(row, col);
      end;
   end;
 end;

begin
 open(srfile, oldfileName('Surface File'));
 for row := 0 to size do
  for col := 0 to size do
   read(srfile, srf[row, col]);
 close(srfile);
 setorigin(-10, -10);
 write('(c)oastline, (s)tripe, (t)opo, (g)rain, or (q)uit?');
 readln(ans);
 repeat
  if ans[1] = 'c' then
   coastdisp
  else if ans[1] = 's' then
   stripedisp
  else if ans[1] = 't' then
   topodisp
  else if ans[1] = 'g' then
   graindisp;
  write('Save?');
  readln(ans);
  if ans[1] = 'y' then
   begin
    write('Filename?');
    readln(ans);
    savedrawing(ans);
   end;
  write('(c)oastline, (s)tripe, (t)opo, (g)rain, or (q)uit?');
  readln(ans);
 until ans[1] = 'q';
end.
```

FRAKFFC.PAS Accompanies "Mimicking Mountains," by Tom Jeffery,
BYTE, December 1987, page 337

```
program fractal_ffc;
 const
  size = 64; (Maximum index of array)
 var
  row, col, n, step, st : longint;
  srf : array[0..size, 0..size] of longint; (The surface file)
  ans : string[10];
  srfile : file of longint;
  H : real; (Roughness factor)
  stepfactor : real;

 function gauss : real;
(Returns a gaussian variable with mean = 0, variance = 1)
(Polar method due to Knuth, vol. 2, pp. 104, 113 )
(but found in "Smalltalk-80, Language and Implementation",)
(Goldberg and Robinson, p. 437.)
  var
   i : integer;
   sum, v1, v2, s : real;
 begin
  sum := 0;
  repeat
   v1 := (random / maxint);
   v2 := (random / maxint);
```

```
    s := sqr(v1) + sqr(v2);
   until s < 1;
   s := sqrt(-2 * ln(s) / s) * v1;
   gauss := s;
  end;

  procedure hordetail (row : longint);
 {Calculates new points for one row}
   var
    disp, i, col : longint;
  begin
   col := 0;
   while col < size do
    begin
     disp := Round(100 * (gauss * stepfactor)); {Random displacement}
     srf[row, col + step] := (srf[row, col] + srf[row, col + 2 * step]) div 2;
 {Midpoint}
     srf[row, col + step] := srf[row, col + step] + disp;{New point}
     col := col + 2 * step;
    end;
  end;

  procedure verdetail (col : longint);
 {Calculates  new points for one column}
   var
    disp, i, row : longint;
  begin
   row := 0;
   while row < size do
    begin
     disp := Round(100 * (gauss * stepfactor)); {Random displacement}
     srf[row + step, col] := (srf[row, col] + srf[row + 2 * step, col]) div 2;
 {Midpoint}
     srf[row + step, col] := srf[row + step, col] + disp; {New point}
     row := row + 2 * step;
    end;
  end;

  procedure centdetail (row : longint);
 {Calculates new points for centers of all cells in a row}
   var
    disp, i, col : longint;
  begin
   col := step;
   while col < size do
    begin
     disp := Round(100 * (gauss * stepfactor)); {Random displacement}
     srf[row, col] := (srf[row, col - step] + srf[row, col + step] + srf[row -
 step, col] + srf[row + step, col]) div 4; {Center Point}
     srf[row, col] := srf[row, col] + disp; {New point}
     col := col + 2 * step;
    end;
  end;

  procedure detail;
 {Calculates new points at current step size}
   var
    i, row, col : longint;
  begin
   row := 0;
   col := 0;
   while row <= size do
    begin
     hordetail(row);
     row := row + 2 * step;
    end;
   while col <= size do
    begin
     verdetail(col);
     col := col + 2 * step;
    end;
   row := step;
   while row <= size - step do
    begin
     centdetail(row);
     row := row + 2 * step;
    end;
 end;

 procedure newsurface;
  var
   row, col : longint;
 begin
  step := size;
```

continued

```
  stepfactor := exp(2 * H * ln(step));
  srf[0, 0] := Round(100 * (gauss * stepfactor));
  srf[0, size] := Round(100 * (gauss * stepfactor));
  srf[size, 0] := Round(100 * (gauss * stepfactor));
  srf[size, size] := Round(100 * (gauss * stepfactor));
  repeat
   step := step div 2; {Go to smaller scale}
   write('step = ');
   writeln(step);
   stepfactor := exp(2 * H * ln(step)); {Factor proportional to step size}
   detail; {Calculate all new points at current step size}
  until step = 1;
 end;

begin
 showtext;
 write('H = ?');{Set roughness}
 readln(H);
 open(srfile, NewfileName('Surface File'));
 st := tickcount;
 randseed := st;  {Randomize}
 newsurface; {Calculate surface}
 for row := 0 to size do
  for col := 0 to size do
   write(srfile, srf[row, col]); {Store surface in file}
 close(srfile);
 st := (tickcount - st) div 3600;
 write(st);
 writeln(' minutes');
end.
```

MAP.PAS Accompanies "Mimicking Mountains," by Tom Jeffery, BYTE, December 1987, page 337

```
program Map;
 const
  size = 64;
  cell = 2;
 type
  twohts = array[1..2] of longint;
  levarray = array[0..8] of longint;
 var
  srf : array[0..size, 0..size] of longint;
  Srfilename, ans : string[30];
  srfile : file of longint;
  col, row, cont : longint;
  Pt, pt2 : point;
  rct : rect;
  Grayscale : array[0..8] of pattern;

 procedure SetUpDrawing;
  var
   R : Rect;
 begin
  HideAll;
  SetRect(R, 0, 38, 511, 341);
  SetDrawingRect(R);
  ShowDrawing;
  setorigin(-30, -30);
  moveto(45, -5);
  drawstring('MAP');
 end;

 function minmax : twohts;
{Min surface height is minmax[1], max is minmax[2]}
  var
   mm : twohts;
   r, c : longint;
 begin
  mm[1] := maxlongint;
  mm[2] := -maxlongint;
  for r := 0 to size do
   for c := 0 to size do
    begin
     if srf[r, c] < mm[1] then
      mm[1] := srf[r, c];
     if srf[r, c] > mm[2] then
      mm[2] := srf[r, c];
    end;
  minmax := mm;
 end;
```

```
 procedure paintpt (row, col : longint);
(Fill a cell x cell square with penpat)
 begin
  pt.v := row * cell;
  pt.h := col * cell;
  pt2.v := (row + 1) * cell;
  pt2.h := (col + 1) * cell;
  rct.topleft := pt;
  rct.botright := pt2;
  paintrect(rct);
 end;

 procedure mapDisp;
  var
   i, sum, row, col, levl, range : longint;
   lev : array[0..8] of longint;
   mm : twohts;
 begin
  mm := minmax;
  range := mm[2] - mm[1];
  sum := mm[1]; (Min height of surface)
  levl := (range div 9) + 1; (Eight height zones, spearated by levl)
  for i := 0 to 8 do
   begin
    sum := sum + levl;
    lev[i] := sum; (Nine height zones, lev[1]-lev[8])
   end;
  moveto(140, -5);
(Legend)
  for i := 0 to 8 do
   begin
    penpat(grayscale[i]);
    pt.h := 140;
    pt.v := i * 15;
    pt2.h := 150;
    pt2.v := i * 15 + 10;
    rct.topleft := pt;
    rct.botright := pt2;
    paintrect(rct);
    penpat(black);
    framerect(rct);
    moveto(153, i * 15 + 10);
    writedraw(lev[i]);
   end;
(Map)
  for row := 0 to size do
   for col := 0 to size do
    begin
     i := 0;
     while (srf[row, col] > lev[i]) do
      i := i + 1; (Compare height to zones)
     penpat(grayscale[i]); (Choose gray shade corresponding to zone)
     paintpt(row, col);
    end;
 end;

begin
 setupdrawing;
(Set up array of patterns)
 grayscale[8] := white;
 grayscale[0] := black;
 stuffhex(@grayscale[7], '8800220088002200');
 stuffhex(@grayscale[6], 'AA00AA00AA00AA00');
 stuffhex(@grayscale[5], 'AA11AA44AA11AA44');
 stuffhex(@grayscale[4], 'AA55AA55AA55AA55');
 stuffhex(@grayscale[3], 'BB55EE55BB55EE55');
 stuffhex(@grayscale[2], 'FF55FF55FF55FF55');
 stuffhex(@grayscale[1], 'FF77FFDDFF77FFDD');
 repeat
  srfilename := oldfileName('Surface File');
  open(srfile, srfilename);
  for row := 0 to size do
   for col := 0 to size do
    read(srfile, srf[row, col]);
  close(srfile);
  mapdisp; (Draw Map)
  showtext;
  write('Save map?');
  readln(ans);
  if ans[1] = 'y' then
   begin
    write('File name?');
    readln(ans);
```

continued

```
    savedrawing(ans);
   end;
  write('Again?');
  readln(ans);
 until ans[1] = 'n';
end.
```

FRAKVOSS.PAS Accompanies "Mimicking Mountains," by Tom Jeffery,
BYTE, December 1987, page 337

```
program fractalVoss;
 const
  size = 64; {Maximum index of array}

 var
  row, col, n, step, st : longint;
  srf : array[0..size, 0..size] of longint; {The surface file}
  ans : string[10];
  srfile : file of longint;
  H : real; {Roughness factor}
  stepfactor : real;

 function gauss : real;
{Returns a gaussian variable with mean = 0, variance = 1}
{Polar method due to Knuth, vol. 2, pp. 104, 113 }
{but found in "Smalltalk-80, Language and Implementation",}
{Goldberg and Robinson, p. 437.}
  var
   i : integer;
   sum, v1, v2, s : real;
 begin
  sum := 0;
  repeat
   v1 := (random / maxint);
   v2 := (random / maxint);
   s := sqr(v1) + sqr(v2);
  until s < 1;
  s := sqrt(-2 * ln(s) / s) * v1;
  gauss := s;
 end;

 procedure horintpol (row : longint);
{Interpolates midpoints for 1 row}
  var
   i, col : longint;
 begin
  col := 0;
  while col < size do
   begin
    srf[row, col + step] := (srf[row, col] + srf[row, col + 2 * step]) div 2;
{New point}
    col := col + 2 * step;
   end;
 end;


 procedure verintpol (col : longint);
{Interpolates midpoints for 1 column}
  var
   i, row : longint;
 begin
  row := 0;
  while row < size do
   begin
    srf[row + step, col] := (srf[row, col] + srf[row + 2 * step, col]) div 2;
{New point}
    row := row + 2 * step;
   end;
 end;

 procedure centintpol (row : longint);
{Interpolates center points for all cells in a row}
  var

   i, col : longint;
 begin
  col := step;
  while col < size do
   begin
```

```
    srf[row, col] := (srf[row, col - step] + srf[row, col + step] + srf[row -
step, col] + srf[row + step, col]) div 4;
(New point)
    col := col + 2 * step;
   end;
 end;

 procedure intpol;
(Interpolates all midpoints at current step size)
  var
   i, row, col : longint;
 begin
  row := 0;
  col := 0;
  while row <= size do
   begin
    horintpol(row);
    row := row + 2 * step;
   end;
  while col <= size do
   begin
    verintpol(col);
    col := col + 2 * step;
   end;
  row := step;
  while row <= size - step do
   begin
    centintpol(row);
    row := row + 2 * step;
   end;
 end;

 procedure detail;
(Adds random displacement to all points at current step size)
  var
   r, c, disp : longint;
 begin
  r := 0;
  while r <= size do
   begin
    c := 0;
    while c <= size do
     begin
      disp := Round(100 * (gauss * stepfactor));
      srf[r, c] := srf[r, c] + disp;
      c := c + step;
     end;
    r := r + step;
   end;
 end;

 procedure newsurface;
 begin
  step := size;
  stepfactor := exp(2 * H * ln(step));
  srf[0, 0] := Round(100 * (gauss * stepfactor));
  srf[0, size] := Round(100 * (gauss * stepfactor));
  srf[size, 0] := Round(100 * (gauss * stepfactor));
  srf[size, size] := Round(100 * (gauss * stepfactor));
  repeat
   step := step div 2; (Go to smaller scale)
   write('step = ');
   writeln(step);
   stepfactor := exp(2 * H * ln(step)); (Factor proportional to step size)
   intpol;
   detail;
  until step = 1;
 end;

begin
 showtext;
 write('H = ?'); (Set roughness)
 readln(H);
 open(srfile, NewfileName('Surface File'));
 st := tickcount;
 randseed := st; (Randomize)
 newsurface; (Calculate surface)
 for row := 0 to size do
  for col := 0 to size do
   write(srfile, srf[row, col]); (Store surface in file)
 close(srfile);
 st := (tickcount - st) div 3600;
 write(st);
```

continued

NLDOS.DOM Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*************************************************************************/
/*
        NLDOS.DOM

        Copyright 1987, Alex Lane

        File 2 of 7.
        This file declares the user-defined domains used in the
        NLDOS program.

*/
/*************************************************************************/

DOMAINS
    worktok     =  token(string);
                   p_token(string);
                   drive(string);
                   directory(string);
                   targetspec(string);
                   sourcespec(string);
                   targetfile(string);
                   filespec(string);
                   command(string);
                   parameter(string);
                   assignment(string)
    worklist    =  worktok *
    stringlist  =  string *
    symbolist   =  symbol *
    charlist    =  char *

/********************** end of file ********************************/
```

NLDOS.SYN Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*************************************************************************/
/*
        NLDOS.SYN

        Copyright 1987, Alex Lane

        File 6 of 7.
        Synonyms for various commands, switches, and modifiers.
        Revision 1.1

*/
/*************************************************************************/


CLAUSES

chaff( ["PLEASE","ME","YOU","THE","A", "DO", "THAT", "ARE", "IS"]).

c_syn( "DEL", ["DELETE", "KILL", "ERASE", "ZAP", "CHOP", "REMOVE"] ).
c_syn( "COPY", ["COPY", "CLONE"]).
c_syn( "CHDIR", ["CHDIR", "CD"]).
c_syn( "REN", ["RENAME" ]).
c_syn( "DIR", ["DIRECTORY","CATALOG", "SEE"]).
c_syn( "TYPE", ["TYPE", "LIST", "VIEW"]).
c_syn( "BACKUP", ["BACKUP"]).
c_syn( "RESTORE", ["RESTORE"]).
c_syn( "PATH", ["PATH"]).
c_syn( "TIME", ["TIME"]).
c_syn( "BREAK", ["BREAK"]).
c_syn( "DATE", ["DATE"]).
```

```
c_syn( "PROMPT", ["PROMPT"]).
c_syn( "SET", ["SET","ENVIRONMENT"]).

f_syn( "ALL", ["ALL", "EVERYTHING", "ENTIRE", "COMPLETELY"]).
f_syn( "SHOW", ["SHOW"] ).
f_syn( "DIRECTORY", ["DIRECTORY"]).
f_syn( "SUBDIRECTORY", ["SUBDIRECTORY"]).
f_syn( "AFTER", ["AFTER", "SINCE"]).
f_syn( "FILE", ["FILES"]).
f_syn( "ON", [ "ON","IN"]).
f_syn( "FROM", ["FROM", "SOURCE"]).
f_syn( "TO", ["TO", "DESTINATION"]).
f_syn( "CHANGE", ["CHANGE","GOTO"]).
f_syn( "MAKE", ["MAKE", "CREATE"]).
f_syn( "CONTENT", ["CONTENTS","LISTING"]).

d_syn( "/W", ["/W", "WIDE", "MULTICOLUMN", "COLUMN"]).
d_syn( "/P", ["/P", "PAUSE", "PAGE" ]).

b_syn( "/S", ["/S", "SUBDIRECTORY"]).
b_syn( "/D:", ["/D:", "AFTER"]).
b_syn( "/M", ["/M", "MODIFIED", "CHANGED"]).
b_syn( "/A", ["/A", "ADD"]).
b_syn( "/P", [ "/P", "PROMPT"]).

/********************** end of file ********************************/
```

NLSIMPLE.DOC Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

NLSIMPLE.PRO: An Experiment in Natural Language Using Turbo Prolog
by Rich Malloy, BYTE Magazine, August 1987

This program is a very simple natural language interface for MS-DOS. For simplicity, it keeps all of its grammar rules in strings, and for that reason, is a bit slow. It also uses a context-free grammar with top-down parsing. This approach involves a large amount of backtracking, and thus can be somewhat inefficeint. The production rules (or rewrite rules) used by the grammar are functionally similar to context-free-grammar rules or to the Definite Clause Grammar (DCG) rules of standard C&M Prolog, but look a little different. For example, DCG rules look like:

```
S -> NP VP
VP -> VERB ADV NP
```

The rules in this program look something like this:

```
rule("S", "NP VP", "NP VP").
rule("VP", "VERB ADV NP", "VERB ADV NP").
rule("S", "help", "Consult the manual").
```

The first argument of the rule is similar to the left side of the DCG rule. The second argument is similar to the right side. As the system executes, it builds an internal representation of the input phrase until that representation matches the input exactly. If the first argument appear in that internal representation, it can be replaced with the second argument. The "S" is merely a start symbol.

The third argument controls the output of the system. Just as an internal representation is being built, an output string is also being built. And, if match is found, and the first argument appears in the output string, then it is replaced by the third argument.

In the above case, if the input were "help", the last rule would allow the internal representation "S" to match the input exactly. Then the output string, which starts as "S", would be replaced by "Consult the manual".

Another difference between this approach and classic context-free grammars is that it uses semantic categories (e.g., ALLFILES, which stands for words such as "everything") instead of word categories such as Noun and Verb.

If you choose the "rundos" mode rather than the "runtest" mode,

continued

the program will send a command to DOS. Unfortunately, Turbo Prolog clears the screen after it finishes a DOS command, making it very hard to read a directory. For that reason, this program sends its DOS commands to a batch file that ends with a pause command. The program then calls the DOS to execute that batch file. This batch file approach would be useful also for operations that require several DOS commands.

As the program works on the input phrase, it displays the words for which it has found a match on the screen. If the program does not understand an input phrase, the first word that is not displayed is probably the one that caused the problem.

The user should feel free to add more capabilities to the system. Commands such as BACKUP, FIND, SORT, etc. can be easily implemented. The user may also want to modify the "equals" predicate so that both uppercase and lower case input is accepted.

Sample input (note all input must be lower case):

```
show all files on b
let me see all files on the main drive
copy all files beginning with x from the main drive to b:
```

When using the "rundos" version of the program, enter your command at the DOS prompt as follows:

```
nlsimple show everything on the main drive
```

To save keystrokes, you may want rename the executable version of the program with a shorter name.

Note that in the program there are some rules for parsing simple mathematical phrases such as:

```
A * B + C * D
```

These rules can be easily expanded to handle more complicated phrases. Refer to most textbooks on AI for examples of how to build a context-free grammar to parse mathematical expressions.

NL-READ.ME Accompanies "DOS in English," by Alex Lane, BYTE, December 1987, page 261

There are seven files for the natural-language-interface to DOS program outlined by Alex Lane in his article in the December 1987 issue of BYTE. These are:

```
NLDOS     PRO
NLDOS     DOM
NLDOS     SYN
NLRULES   PRO
NLTOKENS  PRO
NLDATE    PRO
NLUTILS   PRO
```

To use the program, run Turbo Prolog and load NLDOS.PRO. When you compile this file, it will load all of the other files. The program is currently set up to run in test mode. In this mode, it will not issue commands to DOS, but simply displays the commands for the user's inspection. Follow the instructions in the program and in Turbo Prolog for compiling an executable program that will feed commands to DOS.

Also in this area is a somewhat smaller program called NLSIMPLE.PRO that does similar things using a context-free-grammar approach. Note that the grammar rules are stored as strings, and as such are relatively slow. The strings, however, are easy to modify. See NLSIMPLE.DOC for more details.

NLUTILS.PRO Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*************************************************************************/
/*
        NLUTILS.PRO

        Copyright 1987, Alex Lane

        File 3 of 7.
        Utility predicates for NLDOS.

*/
/*************************************************************************/


PREDICATES

/*  various permutations and combinations of 'member'-like predicates */

member(worktok, worklist)
member(char, charlist)
member(symbol,symbolist)
member(string,stringlist)
member_head(worktok, worklist, worklist)
member_head(symbol,symbolist,symbolist)
member_phrase(worklist, worklist)
member_phrase(stringlist,stringlist)

/* the standard 'append' predicate */

append(worklist,worklist,worklist)
append(charlist,charlist,charlist)
append(symbolist,symbolist,symbolist)

repeat

remove(string,stringlist,stringlist)
remove(worktok,worklist,worklist)
remove_once(worktok,worklist,worklist)
remove_list_once(worklist,worklist,worklist)

CLAUSES

member( X, [X|_] ).
member( X, [_|Y] ) :- member ( X, Y ).

member_head( X, [X|T], T ).
member_head( X, [_|A], T ) :- member_head( X, A, T ).

member_phrase( [], _ ).
member_phrase( [H|T], [H|U] ) :- member_phrase( T, U ).
member_phrase( X, [_|T] ) :- member_phrase( X, T ).


append([],L,L).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).

repeat.
repeat:-repeat.

remove(_,[],[]) :- !.
remove(X,[X|T],F) :- !, remove( X, T, F ).
remove(X,[A|T],[A|F]) :- remove( X, T, F).

remove_once( _, [], [] ) :- !.
remove_once( X, [X|T], T ).
remove_once( X, [A|T], [A|F] ) :- remove_once( X, T, F ).

remove_list_once( [], X, X ) :- !.
remove_list_once( [H|T], L, F ) :- remove_once( H, L, F0 ),
                                   remove_list_once( T, F0, F ).

/********************** end of file ********************************/
```

continued

NLTOKENS.PRO Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*********************************************************************/
/*
        NLTOKENS.PRO

        Copyright 1987, Alex Lane

        File 4 of 7.
        Tokenizer for NLDOS program.

*/
/*********************************************************************/


PREDICATES

backslash(char)
colon(char)
digit(char)
dot_delim(char)
get_token_list(string,worklist)
letter(char,char)
list_text(string, charlist)     /* (i,o) */
list_text1(string,charlist,string)
makeworktok(symbol,string,worktok)
process_in_str(charlist,charlist,charlist,charlist)
rest_token( charlist, charlist, worktok, charlist, symbol)
string_delim(char)
tokenize(charlist,worklist,worklist)


CLAUSES

get_token_list(String,Result) :-
        list_text(String,[Char|Tail]),  trace(on),
        tokenize([Char|Tail], [], Result ).

list_text( "", [] ) :- !.
list_text(String, [H|T] ) :-
        bound(String),
        frontchar(String,H,Rest),
        list_text(Rest,T).

list_text(String, [H|T] ) :-
        bound(H),
        list_text1("",[H|T],String).

list_text1( A, [], A ) :- !.
list_text1( A, [H|T], Out ) :-
        str_char( Hs, H ),
        concat( A, Hs, AHs ),
        list_text1( AHs, T, Out ).

tokenize([H|T],List,L ) :-
        letter(H, Letter),!,    /* can be anything... */
        rest_token(T,[Letter],Word,Rem, token),
        append(List,[Word],Nlist),
        tokenize(Rem,Nlist,L).

tokenize([H|T], List, L ) :-
        backslash(H),!,         /* seems to be a directory */
        rest_token(T,[H],Num, Rem, dir),
        append(List,[Num],Nlist),
        tokenize(Rem,Nlist,L).

tokenize([H|T], List, L ) :-
        dot_delim(H),!,         /* seems to be an incomplete filespec */
        rest_token(T,[H],Num, Rem, fspec),
        append(List,[Num],Nlist),
        tokenize(Rem,Nlist,L).

tokenize([H|T], List, L ) :-
        string_delim(H),!,      /* seems to be a string... */
        rest_token(T,[H],Num, Rem, string),
        append(List,[Num],Nlist),
        tokenize(Rem,Nlist,L).

tokenize(['='|T], List, L ) :-
        append(List,[assignment("=")],Nlist),!,
        tokenize(T,Nlist,L).
```

```
tokenize([_|T], List, L ) :-
        !, tokenize(T,List, L ).

tokenize([],List,List ).

process_in_str([H|T],L,T,L1) :-
        append(L,[H],L1).

/*  As long as you encounter letters in the current token, you
    continue to be what the head of the token modifier list
    says you are.  The one exception is continuing to find letters
    once having established that a drive name has already been
    found.  We are going to ASSUME that these letters belong to a
    file name.
*/

rest_token( [H|T], List, Word, X, string ) :- /* special case?! */
        H <> '\"',H <> '\\',!,
        append( List, [H], Nlist ),
        rest_token( T, Nlist, Word, X, string ).

rest_token( ['\\'|T], List, Word, X, string ) :- /* special case?! */
        append( List, ['\\'], Nlist ),
        process_in_str(T,Nlist,T1,Nlist1),!,
        rest_token( T1, Nlist1, Word, X, string ).

rest_token( [H|T], List, Word, X, drive ) :- /* special case?! */
        letter( H, Letter ),!,
        append( List, [Letter], Nlist ),
        rest_token( T, Nlist, Word, X, fspec ).

rest_token( [H|T], List, Word, X, TL ) :-
        letter( H, Letter ),!,
        append( List, [Letter], Nlist ),
        rest_token( T, Nlist, Word, X,TL ).

rest_token( [H|T], List, Word, X, _ ) :-
        colon( H ),!,           /* have found a drive! */
        append( List, [H], Nlist ),
        rest_token( T, Nlist, Word, X, drive ).

rest_token( [H|T], List, Word, X, _ ) :-
        backslash( H ),!,       /* have found a directory */
        append( List, [H], Nlist ),
        rest_token( T, Nlist, Word, X, dir ).

rest_token( [H|T], List, Word, X, _ ) :-
        dot_delim( H ),!,               /* have found a filespec */
        append( List, [H], Nlist ),
        rest_token( T, Nlist, Word, X, fspec ).

rest_token( [' '|T], List, Word, T, Token ) :-
        list_text( Word0, List ),!,
        makeworktok(Token,Word0,Word).

rest_token( ['\"'|T], List, Word, T, Token ) :-
        append(List,['\"'],List1),
        list_text( Word0, List1 ),!,
        makeworktok(Token,Word0,Word).

rest_token( ['='|T], List, Word, ['='|T], Token) :-
        list_text( Word0, List ),!,
        makeworktok(Token,Word0,Word).

rest_token( [_|T], List, Word, X, TL ) :-
        !, rest_token( T, List, Word, X, TL ).

rest_token( [], List, Word, [], Token ) :-
        !, list_text( Word0, List ),
        makeworktok(Token,Word0,Word).

makeworktok(token,A,token(A)).
makeworktok(fspec,A,filespec(A)).
makeworktok(drive,A,drive(A)).
makeworktok(dir,A,directory(A)).
makeworktok(string,A,parameter(A)).

letter(C, D) :-
        C >= 'a', C <= 'z', !,
        str_char(C1,C),
        upper_lower(D1,C1),
        str_char(D1,D).
letter(C, C) :-
        C >= 'A', C <= 'Z', !;
```

continued

```
        digit(C),!;
        member(C,['_','$','?','*','/','<','>','|','-']).

dot_delim('.').

backslash('\\').
digit(C) :-
        C >= '0', C <= '9'.

colon(':').

string_delim('\"').

/********************** end of file ********************************/
```

NLDOS.DOC Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

This is a transcript of a session with NLDOS, a natural-language interface for DOS written for Turbo Prolog. It shows a series of input statements, and the resulting command generated by the program. Note that the transcript has been formatted for easier reading. The actual transcripts produced by the program should look a little different.

```
>dir

Command:    DIR C:\AI\PROLOG

>wide directory

        Rule fired: establish_dir_flags

Command:    DIR C:\AI\PROLOG /W

>wide directory with pause

        Rule fired: establish_dir_flags
        Rule fired: establish_dir_flags

Command:    DIR C:\AI\PROLOG /W/P

>direc

Command:    DIR C:\AI\PROLOG

>page directory

        Rule fired: establish_dir_flags

Command:    DIR C:\AI\PROLOG /P

>cat

Command:    DIR C:\AI\PROLOG

>catalog

Command:    DIR C:\AI\PROLOG

>directory *.*

Command:    DIR *.*

>show all files

        Rule fired: remove_word_file
        Rule fired: show_file_to_dir

Command:    DIR C:\AI\PROLOG

>copy *.* b:

Command:    COPY *.* C:\AI\PROLOG
```

```
>copy all files to b:

        Rule fired: remove_word_file
        Rule fired: establish_targetspec

Command:    COPY *?.* B:

>copy all files from c: to a:

        Rule fired: remove_word_file
        Rule fired: establish_targetspec
        Rule fired: establish_sourcespec
        Rule fired: establish_sourcespec

Command:    COPY C:*?.* A:

>delete all files on b:

        Rule fired: remove_word_file
        Rule fired: consolidate_drive_filespec

Command:    DEL B:*?.*

>kill all .c files

        Rule fired: remove_word_file
        Rule fired: remove_word_file

Command:    DEL *.C

>zap *.bak

Command:    DEL *.BAK

>zap *.bak files

        Rule fired: remove_word_file

Command:    DEL *.BAK

>restore all files in all subdir

        Rule fired: remove_word_file
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags

Command:    RESTORE A: *?.* /S

>backup all files

        Rule fired: remove_word_file

Command:    BACKUP *?.* A:

>backup all files to b:

        Rule fired: remove_word_file
        Rule fired: establish_targetspec

Command:    BACKUP *?.* B:

>backup all files in all subdir modified after 9/sep

        Rule fired: remove_word_file
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags

Command:    BACKUP *?.* A: /S/D:09-9-87

>backup all modified files to a:

        Rule fired: remove_word_file
        Rule fired: establish_targetspec
        Rule fired: establish_bac_flags

Command:    BACKUP *?.* A: /M
```

continued

```
>backup all modified .c files to a:

        Rule fired: remove_word_file
        Rule fired: remove_word_file
        Rule fired: establish_targetspec
        Rule fired: establish_bac_flags

Command:    BACKUP *.C A: /M

>backup all .c modified in subdir to b:

        Rule fired: remove_word_file
        Rule fired: consolidate_drive_filespec
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags

Command:    BACKUP B:*.C A: /M/S

>backup foo.* to a: in all subd modif after 1-jan-87

        Rule fired: establish_targetspec
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags
        Rule fired: establish_bac_flags

Command:    BACKUP FOO.* A: /S/D:01-1-87

>what time is it

Command:    TIME

>what time

Command:    TIME

>time

Command:    TIME

>show environ

Command:    SET

>prompt

Command:    PROMPT

>show prompt

Command:    PROMPT

>show content of foo.bar

Rule fired: show_content_to_type

Command:    TYPE FOO.BAR

>ren foo.bar to sap.foo

Rule fired: establish_targetfile

Command:    Incomplete RENAME command.

>rename foo.bar sap.foo

Command:    Incomplete RENAME command.

>show environment

Command:    SET

>environ

Command:    SET

>env

Command:    SET

>set

Command:    SET
```

```
>what date

Command:   DATE

>show date

Command:   DATE

------- End ---------
```

NLDATE.PRO Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*********************************************************************/
/*
        NLDATE.PRO

        Copyright 1987, Alex Lane

        File 5 of 7.
        Date finder.  Given a token with a date embedded and
        delimited with "-" or "/", takes a best shot at extracting
        a date.  Finds year first, then month and day.  European
        style (6-1-87 for January 6, 1987) fares poorly.

        Revision 1.2 (9/09/87)

*/
/*********************************************************************/

PREDICATES

build_date(string,string)
find_day(stringlist,stringlist,string)
find_month(stringlist,stringlist,string)
find_year(stringlist,stringlist,string)
get_parts(string,stringlist)
month(string,string)
normalize_year(integer,integer)

CLAUSES

build_date(Instring,Outstring) :-
        get_parts(Instring,Intokens),
        find_year(Intokens,O1,Year),
        find_month(O1,O2,Month),
        find_day(O2,_,Day),
        concat(Month,"-",MP),
        concat(Day,"-",DP),
        concat(MP,DP,MDP),
        concat(MDP,Year,Outstring).

get_parts("",[]).
get_parts( In, [D|SubOut] ) :-
        fronttoken(In,D,Rest),
        get_parts(Rest,SubOut).

find_year(In,Out,Y ) :-
        member(X, In ),
        str_int(X,X1),
        normalize_year(X1,X2),
        X2 >= 80,!,
        str_int(Y,X2),
        remove(X,In,Out).
find_year(X,X,"87").

normalize_year( In,Out) :-
        In >= 1980,!,
        Out = In - 1900.
normalize_year(X,X) :- !.

find_month( In, Out, M ) :-
        month( M, Month ),
        str_len( Month, Mlen ), Mlen >= 3,
        member( X, In ),
        concat( X,_,Month),
        remove( X, In, Out).
```

continued

```
find_month( In, Out, M ) :-
        member(X, In ),
        str_int(X,X1),
        X1 > 0,
        X1 < 13,
        str_int(M,X1),
        remove(X,In,Out).

find_day( In, Out, M ) :-
        member(X, In ),
        str_int(X,X1),
        X1 > 0,
        X1 < 32,        /* we don't do the "Thirty days hath..." bit */
        str_int(M,X1),
        remove(X,In,Out).

month( "01", "JANUARY").
month( "02", "FEBRUARY").
month( "03", "MARCH").
month( "04", "APRIL").
month( "05", "MAY").
month( "06", "JUNE").
month( "07", "JULY").
month( "08", "AUGUST").
month( "09", "SEPTEMBER").
month( "10", "OCTOBER").
month( "11", "NOVEMBER").
month( "12", "DECEMBER").

/********************** end of file ********************************/
```

NLRULES.PRO Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*********************************************************************/
/*
        NLRULES.PRO

        Copyright 1987, ALex Lane

        File 7 of 7.
        Rules used to massage the token list in NLDOS.
        Revision 1.1

*/
/*********************************************************************/
PREDICATES

rule(symbol,worklist,worklist)


CLAUSES

rule( show_file_to_dir, In, Out) :-
        repl( [token("SHOW"),filespec("*?.*")], command("DIR"), In, Out ).
rule( show_content_to_type, In, Out ) :-
        repl( [token("SHOW"),token("CONTENT")],command("TYPE"), In, Out).
rule( dos_directory_commands, In, Out ) :-
        repl( [token("CHANGE"),command("DIR ")],command("CHDIR"), In, Out);
        repl([command("DIR "),token("PATH")],command("PATH"), In, Out).
rule( remove_word_file, Input,Output) :-
        member( token("ALL"), Input),
        member( filespec(A), Input),
        E = [token("ALL"),filespec(A)],
        concat("*",A,A1),
        member_phrase( E, Input),
        repl( E, filespec(A1), Input, Output);
        member(filespec(A),Input),
        repl([token("ALL"),filespec(A)],filespec(A),Input,Output);
        repl( [token("ALL"),token("FILE")], filespec("*?.*"),Input,Output);
        member(filespec(A),Input),
        repl([filespec(A),token("FILE")],filespec(A),Input,Output).
rule( consolidate_drive_filespec, Input,Output) :-
        member( filespec(A), Input),
        member( drive(B), Input ),
        concat(B,A,C),
        E = [filespec(A),token("ON"),drive(B)],
        repl(E, filespec(C), Input, Output);
        member( filespec(A), Input),
        member( directory(B), Input ),
```

```
        concat(B,A,C),
        E = [filespec(A),token("ON"),directory(B)],
        repl(E, filespec(C), Input, Output).
rule( establish_targetspec, Input,Output) :-
        member(drive(B),Input),
        repl([token("TO"),drive(B)],targetspec(B),Input,Output);
        member(directory(B),Input),
        repl([token("TO"),directory(B)],targetspec(B),Input,Output).
rule( establish_sourcespec, Input,Output) :-
        member(drive(B),Input),
        repl([token("FROM"),drive(B)],sourcespec(B),Input,Output);
        member(directory(B),Input),
        repl([token("FROM"),directory(B)],sourcespec(B),Input,Output);
        member_phrase([filespec(A),sourcespec(B)],Input),
        concat(B,A,C),
        repl([filespec(A),sourcespec(B)],sourcespec(C),Input,Output).
rule(establish_targetfile, Input, Output) :-
        member(filespec(A),Input),
        repl([token("TO"),filespec(A)],targetfile(A),Input,Output).
rule( establish_dir_flags, Input, Output) :-
        member( command("DIR"), Input),
        d_syn(X,Y),
        checkout( Y, Input, Test), !,
        replace_tokens([token(Test)],parameter(X),Input,Output).
rule( establish_bac_flags, Input, Output):-
        is_backup_or_restore(Input),
        b_syn(X,Y),
        checkout( Y, Input, Test ), !,
        replace_tokens([token(Test)],parameter(X),Input,Output);
        repl([parameter("/M"),parameter("/D:")],parameter("/D:"),Input,Output);
        repl([token("ALL"),parameter("/S")],parameter("/S"),Input,Output);
        member(parameter("/D:"),Input),!,
        member(token(A),Input),
        build_date(A,C),
        concat( "/D:",C,DC),
        replace_tokens([parameter("/D:")],parameter(DC),Input,Output).
rule( find_filespec, Input, Output ) :-
        not ( member( filespec(_), Input ) ),
        replace_tokens([token("ALL")],filespec("*?.*"),Input,Output).
rule( out_of_rules, X, X ) :- !.
rule( _, X, X ) :- !.

/********************** end of file ********************************/
```

NLSIMPLE.PRO Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*************************************************************

  NLSIMPLE.PRO: An Experiment in Natural Language Using Turbo Prolog
  by Rich Malloy, BYTE Magazine, August 1987

  This program is a very simple natural language interface for
  MS-DOS. For simplicity, it keeps all of its information in strings,
  and for that reason, is a bit slow. See NLSIMPLE.DOC for more
  details.

  Sample input (note all input must be lower case):

     show all files on b
     let me see all files on the main drive
     copy all files beginning with x from the main drive to b:

  For the sake of file safety, no format, delete, or erase commands
  have been implemented.


*************************************************************/


domains
   symlist = symbol*
   file = batfile
database
   lastword(integer)     /* global var - points to last processed word */
predicates
   equals(string, string)
   matches(string, string, string, string, integer)
   rule(string, string, string)
```

continued

```
    process(string, string)
    sub(string, string, string, string)
    add_lead_space(string, string, string)
    remove_space(string, string)
    display_word(string, integer)
    runtest
    rundos

goal

    runtest.    /* for testing the program */
/*
    rundos.     /* for controlling DOS */
*/

clauses

    runtest if
       asserta( lastword(0) ), !,              /* set global variable */
       makewindow(1,7,7,"NLSIMPLE",0,0,18,79),  /* set up window */
       write("?>"),                            /* write prompt */
       readln(Input),
       process(Input, _),
       nl.

    rundos if
       asserta( lastword(0) ), !,              /* set global variable */
       comline(Input),
       process(Input, Output),
       openwrite(batfile, "NLSIMP99.BAT"),
       writedevice(batfile),
       write(Output), nl,
       write("pause"), nl,
       closefile(batfile),
       system("NLSIMP99").

    process(Input,Output) if
       nl,
       write("Processing:"), nl,
       matches(Input, "S", "S", Temp, 1),    /* "S" = Start symbol */
       remove_space(Temp, Output),
       nl, nl,
       write("Command:"), nl,
       write(Output).
    process(Input, Output) if
       nl, nl, write("Sorry, I don't understand: '", Input, "'"),
       Output = "  ".


/* Rewrite rules  -  specifies how "S" can be rewritten */

    /* The first argument of 'rule' can be rewritten as the second,
          and, if a match is found with the input, any occurence
          of the first argument in the output string should be
          rewritten as the third in the output.

       "!" is used to temporarily separate words that will
            later be adjoining.

      Abbreviations:
           S          Sentence or Start symbol
           VP         Verb phrase
           Q          Question, such as, what time is it?
           H          Help phrase
           SHOW       Show Verb
           COPYV      Copy verb
           DET        Determiner, i.e., "the"
     */


    rule("S", "VP", "VP").
    rule("S", "Q", "Q").
    rule("S", "H", "(Enter any phrase such as: let me see all files)").

    rule("VP", "SHOW SOMEFILES ONDRIVE", "DIR ONDRIVE!SOMEFILES").
    rule("VP", "SHOW ALLFILES ONDRIVE SOMEFILES", "DIR ONDRIVE!SOMEFILES").
    rule("VP", "COPYV SOMEFILES ONDRIVE TODRIVE",
       "COPY ONDRIVE!SOMEFILES TODRIVE").
    rule("VP", "COPYV SOMEFILES TODRIVE ONDRIVE",
       "COPY ONDRIVE!SOMEFILES TODRIVE").
    rule("VP", "RENAMEV SOMEFILE NEWNAME", "RENAME SOMEFILE NEWNAME").
    rule("VP", "change the name of SOMEFILE to NEWNAME",
       "RENAME SOMEFILE NEWNAME").
```

```
   rule("SHOW", "show", "").
   rule("SHOW", "show me", "").
   rule("SHOW", "let me see", "").
   rule("SHOW", "dir", "").
   rule("SHOW", "catalog", "").

   rule("COPYV", "copy", "").
   rule("RENAMEV", "rename", "").

   rule("ALLFILES", "all files", "").
   rule("ALLFILES", "everything", "").
   rule("ALLFILES", "all", "").
   rule("ALLFILES", "", "").

   rule("SOMEFILES", "ALLFILES FILECONDITION", "FILECONDITION").
   rule("SOMEFILES", "SOMEFILE", "SOMEFILE").
   rule("SOMEFILE", "Wildcard*.*", "Wildcard*.*").
   rule("SOMEFILE", "Wildcard.*", "Wildcard.*").
   rule("SOMEFILE", "Wildcard", "Wildcard").
   rule("SOMEFILE", "Wildcard.EXT", "Wildcard.EXT").
   rule("SOMEFILE", "Wildcard:SOMEFILE", "Wildcard.EXT").
   rule("EXT", "Wildcard", "Wildcard").
   rule("NEWNAME", "as SOMEFILE", "SOMEFILE").
   rule("NEWNAME", "SOMEFILE", "SOMEFILE").

   rule("FILECONDITION", "beginning with Wildcard", "Wildcard*.*").
   rule("FILECONDITION", "starting with Wildcard", "Wildcard*.*").
   rule("FILECONDITION", "", "*.*").

   rule("ONDRIVE", "on DRIVE", "DRIVE").
   rule("ONDRIVE", "from DRIVE", "DRIVE").
   rule("ONDRIVE", "DRIVE", "DRIVE").
   rule("ONDRIVE", "", "").
   rule("TODRIVE", "to DRIVE", "DRIVE").
   rule("TODRIVE", "DRIVE", "DRIVE").

   rule("Q", "what TIMEDATE is it", "TIMEDATE").
   rule("Q", "what is the TIMEDATE", "TIMEDATE").
   rule("Q", "what's the TIMEDATE", "TIMEDATE").
   rule("Q", "what is today", "DATE").
   rule("Q", "what is today's date", "DATE").
   rule("TIMEDATE", "time", "TIME").
   rule("TIMEDATE", "date", "DATE").

   rule("H", "help", "").
   rule("H", "help me", "").
   rule("H", "?", "").
   rule("H", "what can you do", "").

   rule("DRIVE", "a:", "a:").
   rule("DRIVE", "a", "A:").
   rule("DRIVE", "a:", "A:").
   rule("DRIVE", "b", "B:").
   rule("DRIVE", "b:", "B:").
   rule("DRIVE", "c:", "C:").
   rule("DRIVE", "c", "C:").
   rule("DRIVE", "d:", "D:").
   rule("DRIVE", "d", "D:").
   rule("DRIVE", "DET main drive", "C:").
   rule("DRIVE", "DET main disk", "C:").

   rule("DET", "", "").
   rule("DET", "the", "").


/* Mathematical rewrite rules:
    These rules can be used to evaluate expressions such as
       A*B+C*D
*/

/*
   rule("S", "Expr", "S(Expr)").
   rule("Expr", "Expr1 + Expr", "plus(Expr1, Expr)").
   rule("Expr", "Expr1", "Expr1").
   rule("Expr1", "Expr2 * Expr1", "mult(Expr2, Expr1)").
   rule("Expr1", "Expr2", "Expr2").
   rule("Expr2", "Var", "Var").
   rule("Var", "A", "var(A)").
   rule("Var", "B", "var(B)").
   rule("Var", "C", "var(C)").
   rule("Var", "D", "var(D)").
*/
```

continued

```
/* internal clauses */
/*******************/

   equals(A, A) if !.

/* displays last word that has been successfully matched */

   display_word(String, Num) if
      lastword(Lastnum),
      Num > Lastnum,
      fronttoken(String, Word, _),
      add_lead_space(String, Word, Word2),
      retract( lastword(_) ),
      asserta( lastword(Num) ),
      lastword(Num),
      write(Word2), !.
   display_word(_, _).

/* Matches: if first argument matches second, then fine. If not,
   clause searches for a substitution in the second such that the
   second will then match the first. The third and fourth are the
   initial and final output strings. The fifth argument points to
   the number of the word in the input string that is being processed */

   matches("", "", X, X, _).
   matches(S1, S2, R1, R2, N) if   /* heads of S1 and S2 match */
      fronttoken(S1, S1head, S1tail),
      fronttoken(S2, S1head, S2tail),
      N2 = N + 1,
      display_word(S1, N2),
      matches(S1tail, S2tail, R1, R2, N2).
   matches(S1, S2, R1, R2, N) if   /* heads of S2 is a wildcard */
      fronttoken(S1, S1head, S1tail),
      fronttoken(S2, "Wildcard", S2tail),
      N2 = N + 1,
      display_word(S1, N2),
      sub(R1, "Wildcard", S1head, R3),
      matches(S1tail, S2tail, R3, R2, N2).
   matches(S1, S2, R1, R2, N) if    /* heads of S1 and S2 can match */
      fronttoken(S2, S2head, S2tail),
      rule(S2head, New_S2head, Newer_S2head),
      sub(R1, S2head, Newer_S2head, R3),
      concat(New_S2head, S2tail, New_S2),
      matches(S1, New_S2, R3, R2, N).


/* Substitution clauses */

/* Remove spaces: the second argument is similar to the first, but all
    "!" have been deleted. */

   remove_space(S1, S2) if     /* if S1 begins with ! then close up */
      fronttoken(S1, "!", S1tail),
      remove_space(S1tail, S3),
      add_lead_space(S1, S3, S2), !.
   remove_space(S1, S2) if     /* if not then don't change it */
      fronttoken(S1, S1head, S1tail),
      remove_space(S1tail, S3),
      concat(S1head, S3, S4),
      add_lead_space(S1, S4, S2), !.
   remove_space(_, "").  /* if first is space or empty, then end loop */

/* Substitution clauses: if second argument is found in
      first, then third is substituted, yielding fourth */

   sub("", _, _, "") if !.    /* if first term empty, stop recursion */
   sub(S1, S2, S3, S4) if     /* first word of S1 matches S2 */
      fronttoken(S1, S1head, S1tail),
      S1head = S2,
      sub(S1tail, S2, S3, S5),
      concat(S3, S5, S6),
      add_lead_space(S1, S6, S4), !.
   sub(S1, S2, S3, S4) if     /* first word doesn't match S2 */
      fronttoken(S1, S1head, S1tail),
      sub(S1tail, S2, S3, S5),
      concat(S1head, S5, S6),
      add_lead_space(S1, S6, S4).

/* Add leading spaces: if the first argument begins with spaces,
      then the third is the concatenation of those spaces with the
      second argument */

   add_lead_space(S1, S2, S2) if
      frontchar(S1, Poss_space, _),
      not(char_int(Poss_space, 32)), !.
```

```
    add_lead_space(S1, S2, S3) if       /* if space at front of S1, add */
       frontchar(S1, Poss_space, S1tail),  /* add space to front of S2 */
       char_int(Poss_space, 32),           /* to yield new S3 */
       concat(" ", S2, S4),                /* recursive until no space */
       add_lead_space(S1tail, S4, S3).

/******* End program **************/
```

NLDOS.PRO Accompanies "DOS in English," by Alex Lane,
BYTE, December 1987, page 261

```
/*****************************************************************/
/*
        NLDOS.PRO

        Copyright 1987, Alex Lane

        This program analyzes English-like command-line input and
        tries to develop the equivalent DOS command with correct
        parameters.

        The goal for experimentation in the Turbo Prolog environment
        does not actually issue the command to DOS.  The goal that
        takes advantage of Turbo Prolog's comline/1 predicate
        actually issues the command and should be used with this
        knowledge.  The author cannot accept responsibility for
        lost or damaged files resulting from the operation of this
        program.

        File 1 of 7.
        Revision 1.2 (9/09/87)

        Sample input:

           show all files on b:
           copy everything to a:
           copy nl*.* to b:
           what time is it
           what is today's date
           copy everything from c: to b:

        When using the "rundos" version of the program,
        enter your command at the DOS prompt as follows:

           nldos show all files on b:


*/
/*****************************************************************/

code=3000
include "nldos.dom"

DATABASE
b_syn( string, stringlist )  /* backup/restore synonyms */
c_syn( string, stringlist )  /* command synonyms */
d_syn( string, stringlist )  /* directory flags */
f_syn( string, stringlist )  /* flag synonyms */
chaff( stringlist )

PREDICATES

catenate_2(string,string,string,string)
catenate_3(string,string,string,string,string)
checkout(stringlist,worklist,string)
command_params(string,worklist,string)
cull_chaff(worklist,worklist)
find_command(worklist,worklist)
gather_flags(worklist,string)
get_backup_source(worklist,string,worktok)
get_backup_target(worklist,string)
get_copy_source(worklist,string,worktok)
get_copy_target(worklist,string)
get_dir_target(worklist,string)
get_restore_source(worklist,string)
get_restore_target(worklist,string)
is_backup_or_restore(worklist)
```

continued

```
make_command(worklist,string)
massage(worklist,worklist)
reform(worklist,worklist)
repl(worklist,worktok,worklist,worklist)
replace_tokens(worklist,worktok,worklist,worklist)
runtest
rundos
standardize_words(worklist,worklist)
include "nlutils.pro"
include "nltokens.pro"
include "nldate.pro"

GOAL

     runtest.       /* to test the program */
/*
     rundos.        /* to issue real DOS commands */
*/

include "nldos.syn"
include "nlrules.pro"

CLAUSES

/* goal for experimentation within the Turbo Prolog environment */

   runtest if
        makewindow(1,7,7,"",0,0,15,80),
        repeat,
        write(">"),
        readln(In),
        get_token_list(In, T),
        cull_chaff(T,T0),
        standardize_words(T0,T1),
        find_command(T1,B),
        massage(B,C),
        make_command(C,Command),
        write("Command:   ",Command),nl,
        fail.

/* goal for execution from DOS prompt */

   rundos if
        comline(In),
        get_token_list(In,T),
        cull_chaff(T,T0),
        standardize_words(T0,T1),
        find_command(T1,B),
        massage(B,C),
        make_command(C,Command),
        write("Command:  ",Command),nl,
/*************************************************/
/*      system(Command),                         */
/*************************************************/
/*                                               */
/*  WARNING! Do not uncomment the system() call  */
/*  unless you want the program to actually      */
/*  perform the command on your computer.        */
/*                                               */
/*************************************************/
        write("Your wish is my command  "), nl.

catenate_3(B,C,D,E,F) :-
        concat(B,C,B0),
        concat(B0," ",BC),
        catenate_2(BC,D,E,F).
catenate_2(C,D,E,F) :-
        concat(C,D,C0),
        concat(C0," ",CD),
        concat(CD,E,F).


standardize_words(Input,Output) :-
        f_syn(X,Y),
        checkout(Y,Input,Test),!,
        replace_tokens([token(Test)],p_token(X),Input,Out),
        standardize_words(Out,Output),!.
standardize_words(Input,Output) :-
        reform(Input,Output).

cull_chaff(Input,Output) :-
        chaff(Y),
        checkout(Y,Input,Test),!,
```

```
        remove(token(Test),Input,Out),
        cull_chaff(Out,Output).
cull_chaff(X,X) :- !.

reform(Input,Output) :-
        member(p_token(X),Input),
        replace_tokens([p_token(X)],token(X),Input,Out),
        reform(Out,Output).
reform(X,X) :- !.

find_command(Input,Output) :-
        c_syn(X,Y),
        checkout(Y,Input,Test),!,
        replace_tokens([token(Test)],command(X),Input,Output).
find_command(X,X) :- !.

massage(In, Out) :-
        rule(A,In,Out0),
        A <> "out_of_rules",
        write("Rule fired: ",A),nl,  /* this line may be commented out
                                        for aesthetic purposes */
        massage(Out0,Out),!.

massage(X, X ) :- !.

checkout(Y,Input,Test) :-
        member(Z,Y),
        member(token(Test),Input),
        str_len(Test,Testlen),Testlen >=2,
        concat(Test,_,Z),!.


replace_tokens( [R|S], Replacement, Origlist, NewList ) :-
        remove_list_once(S, OrigList, IntList ),
        member_head( R, Intlist, T ),
        append( H, [R|T], Intlist ),
        append( H, [Replacement|T], NewList ).

repl( Phrase, Replacement,Input, Output ) :-
        member_phrase(Phrase,Input),
        replace_tokens(Phrase,Replacement,Input,Output),!.

is_backup_or_restore(Input) :-
        member( command( "BACKUP" ), Input);
        member( command( "RESTORE" ), Input).

get_copy_source(Input,D,sourcespec(D)) :-
        member( sourcespec(D), Input ),!.
get_copy_source(Input,D,filespec(D)) :-
        member( filespec(D),Input).

get_copy_target(Input, D) :-
        member( targetspec(D), Input),!;
        member( filespec(D),Input),!;
        disk(D).

get_dir_target(Input, D) :-
        member( targetspec(D), Input),!;
        member( filespec(D), Input),!;
        member( directory(D), Input),!;
        member( drive(D), Input),!;
        disk(D).

get_backup_target( Input, D) :-
        member( targetspec(D), Input ),!;
        member( drive(D), Input),!;
        D= "A:".        /* default */

get_backup_source( Input, D, sourcespec(D)) :-
        member( sourcespec(D), Input),!.
get_backup_source( Input, D, filespec(D)) :-
        member( filespec(D), Input).

get_restore_source( Input, D) :-
        member(sourcespec(D),Input),!;
        member(drive(D),Input),!;
        D = "A:".

get_restore_target(A,B) :- get_copy_target(A,B).

gather_flags( Input, D) :-
        member( parameter(S), Input ),
        remove( parameter(S), Input, Input0),
```

continued

```
        gather_flags( Input0, C),
        concat(B,C,D).
gather_flags(_,"") :- !.

make_command(Input,Command) :-
        write(Input),nl,
        member( command(A), Input), !,
        command_params( A, Input, Command ).

command_params( "TIME", _, "TIME" ).
command_params( "PATH", _, "PATH" ).
command_params( "BREAK", _, "BREAK" ).
command_params( "DATE", _, "DATE" ).
command_params( "PROMPT", _, "PROMPT").
command_params( "SET", _, "SET").

command_params( "DEL", Input, Outstring ) :-
        member( filespec(B), Input ),
        concat( "DEL ", B, Outstring ),!;
        Outstring = "Incomplete DELETE command.".

command_params( "COPY", Input, Outstring ) :-
        get_copy_source( Input,A,B),
        member( B, Input ),     /* we NEED a filespec */
        remove( filespec(A), Input, Input1),
        get_copy_target( Input1, Target ),      /* either a target or default */
        catenate_2( "COPY ",A, Target, Outstring),!;
        Outstring = "Incomplete COPY command.".


command_params( "CHDIR", Input, Outstring ) :-
        member( targetspec(A), Input ),
        concat( "CHDIR ", A, Outstring ),!;
        Outstring = "Incomplete CHDIR command.".

command_params(  "REN", Input, Outstring ) :-
        member(filespec(A), Input),
        member(targetfile(B), Input),
        catenate_2( "REN ",A,B, Outstring),!;
        member(filespec(A), Input),
        member(filespec(B), Input),
        B <> A,
        catenate_2( "REN ",A,B, Outstring),!;
        Outstring = "Incomplete RENAME command.".

command_params( "TYPE", Input, Outstring ) :-
        member(filespec(A),Input),
        concat( "TYPE ", A, Outstring),!;
        Outstring = "Incomplete TYPE command.".

command_params( "DIR", Input, Outstring ) :-
        get_dir_target( Input, Target),
        gather_flags(Input,Flags),
        catenate_2( "DIR ", Target, Flags, Outstring ),!;
        Outstring = "Incorrect DIRECTORY command.".

command_params( "BACKUP", Input, Outstring ) :-
        get_backup_source( Input, Source, S),
        remove(S,Input,Input1),
        get_backup_target(Input1,Target),
        gather_flags(Input,Flags),
        catenate_3( "BACKUP ",Source,Target,Flags, Outstring),!;
        Outstring = "Incorrect BACKUP command.".

command_params( "RESTORE", Input, Outstring ) :-
        get_restore_source( Input, Source ),
        get_restore_target( Input, Target ),
        gather_flags( Input, Flags),
        catenate_3( "RESTORE ",Source,Target,Flags, Outstring),!;
        Outstring = "Incorrect RESTORE command.".

/********************** end of file ********************************/
```

SCROLLZOOM.C, Contributed by Jim Kent. Accompanies BYTE's review of the Atari Mega 4, December 1987, page 153

```
/* scrollzoom.c - a little test program to excercise the blitter's
   smudge bit to do a zoom in lo-res */

#include <osbind.h>
```

```
scroll_zoom_screen() /* blow up screen and scroll up and down it */
(
static char sbuffer[32000];
char *screen;
int i;

screen = Physbase();
copy_screen(screen, sbuffer); /* make copy of screen for source */
/* scroll down left side */
for (i=0; i<150; i++)
zoomblit(sbuffer+160*i, screen, 50);
/* scroll up left side */
for (i=150; i>=0; --i)
zoomblit(sbuffer+160*i, screen, 50);
/* scroll down right side */
for (i=0; i<150; i++)
zoomblit(sbuffer+160*i+80, screen, 50);
/* scroll up right side */
for (i=150; i>=0; --i)
zoomblit(sbuffer+160*i+80, screen, 50);
)

main()
(
int blt_status;

if (Getrez() != 0)
        (
        puts("can only work on low res screens");
        exit(0);
        )
blt_status = _xbios(64,-1);     /* inquire blitter state */
if ((blt_status&3) == 3)
        (
        scroll_zoom_screen();
        )
else
        (
        if (blt_status&2)
                puts("Blitter not active on Desktop");
        else
                puts("No blitter installed");
        exit(0);
        )
)
```

ZOOMBLIT.ASM, Contributed by Jim Kent. Accompanies BYTE's review of the Atari Mega 4, December 1987, page 153

```
        public _zoomblit

        ;_zoomblit
        ;zoomblit(source, dest, linecount)
        ;       does a 4x zoom in lo-res with the blitter.
        ;       dest should point to a 32K screen
        ;       source and dest must not overlap.
        ;
_zoomblit
zbregs  reg     a2/d2   ; preserve registers trashed except d0-1 a0-1
firstp  set     4+4*2   ; offset to first parameter
source  set     firstp
dest    set     firstp+4
linecount set   firstp+8

        movem.l zbregs,-(sp)
        lea     zbsource(pc),a2
        move.l  source(sp),(a2)+
        move.l  dest(sp),(a2)+
        move.w  linecount(sp),(a2)+
        pea     zoomblit
trap1438
        move.w  #38,-(sp)
        trap    #14
        addq    #6,sp
        movem.l (sp)+,zbregs
        rts
```

continued

```
zbsource        dc.l    0
zbdest          dc.l    0
zbcount         dc.w    0
pat4    dc.w $0000,$000f,$00f0,$00ff,$0f00,$0f0f,$0ff0,$0fff
        dc.w $f000,$f00f,$f0f0,$f0ff,$ff00,$ff0f,$fff0,$ffff

; BLiTTER BASE ADDRESS

BLiTTER         equ     $FF8A00

; BLiTTER REGISTER OFFSETS

Halftone        equ     0
Src_Xinc        equ     32
Src_Yinc        equ     34
Src_Addr        equ     36
Endmask1        equ     40
Endmask2        equ     42
Endmask3        equ     44
Dst_Xinc        equ     46
Dst_Yinc        equ     48
Dst_Addr        equ     50
X_Count         equ     54
Y_Count         equ     56
HOP             equ     58
OP              equ     59
Line_Num        equ     60
Skew            equ     61


fLineBusy       equ     7
fLineHog        equ     6
fLineSmudge     equ     5

mHOP_Source     equ     $02
mHOP_Halftone   equ     $01

mSkewFXSR       equ     $80
mSkewNFSR       equ     $40

mLineBusy       equ     $80
mLineHog        equ     $40
mLineSmudge     equ     $20

zoomblit

        ; find the blitter
        move.l  #BLiTTER,a1
        move.l  a1,a2

        ;stuff the pattern buffer
        lea     pat4(pc),a0

        move.l  (a0)+,(a2)+     ;stuff 16 words of pattern buffer with
        move.l  (a0)+,(a2)+     ;the zoom 4x pattern
        move.l  (a0)+,(a2)+
        move.l  (a0)+,(a2)+

        move.l  (a0)+,(a2)+
        move.l  (a0)+,(a2)+
        move.l  (a0)+,(a2)+
        move.l  (a0)+,(a2)+

        ;all end masks on always...
        move.w  #-1,d0
        move.w  d0,Endmask1(a1)
        move.w  d0,Endmask2(a1)
        move.w  d0,Endmask3(a1)

        lea     Line_Num(a1),a0 ; point a0 to "on" switch
        move.w  #7,d0           ; load in busy bit#

        ; now fetch source dest and linecount parameters.  Pc relative
        ; so can cope with no a5 global data pointer inside software interrupt
        move.l  zbsource(pc),d1
        move.l  zbdest(pc),a2
        move.w  zbcount(pc),d2
        add.w   #1,d2           ; d2 was line-count minus 1


        ; set up parameters for initial 16 blits.  These are grouped in
        ; four sets of four.  Each set takes care of one bit-plane.
        ; Within a set each blit will take a nibble of the source to
```

```
        ; a word in the dest.
        move.w  #8,Src_Xinc(a1)         ; hit every word of source plane
        move.w  #120+8,Src_Yinc(a1)
        move.w  #32,Dst_Xinc(a1)        ; every 4th word of dest plane
        move.w  #32+(160*3),Dst_Yinc(a1)
        move.w  #5,X_Count(a1)
        move.b  #mHOP_Halftone,HOP(a1)
        move.b  #3,OP(a1)

        ;now go do the zoom in the x direction
        bsr     zplane
        add.l   #2,d1
        adda    #2,a2
        bsr     zplane
        add.l   #2,d1
        adda    #2,a2
        bsr     zplane
        add.l   #2,d1
        adda    #2,a2
        bsr     zplane

        ;at this point every 4th line of the dest screen has been zoomed.
        ;we want to replicate lines.  I'll do this in 3 steps.  Copy line
        ;0 to line 1, then line1 to line2, and last line 2 to line 3.

        move.w  #2,Src_Xinc(a1) ;do copy as if single-bitplane image in one pass
        move.w  #480+2,Src_Yinc(a1) ; skip 3 lines
        move.w  #2,Dst_Xinc(a1)
        move.w  #480+2,Dst_Yinc(a1)
        move.w  #80,X_Count(a1)
        move.b  #mHOP_Source,HOP(a1)
        move.b  #3,OP(a1)
        move.b  #0,Skew(a1)

        suba    #6,a2
        bsr     rpl_line
        bsr     rpl_line
        bsr     rpl_line
        rts


        ; take a nibble to a word 4 times to cover all nibbles in
        ; source word.
zplane
        move.l  d1,Src_Addr(a1)
        move.b  #0,Skew(a1)
        adda    #24,a2
        move.l  a2,Dst_Addr(a1)
        move.w  d2,Y_Count(a1)
        move.b  #mLineBusy+mLineSmudge,(a0)
restart1
        bset.b  d0,(a0)         ; see if busy
        nop
        bne             restart1

        move.l  d1,Src_Addr(a1)
        move.b  #4,Skew(a1)
        suba    #8,a2
        move.l  a2,Dst_Addr(a1)
        move.w  d2,Y_Count(a1)
        move.b  #mLineBusy+mLineSmudge,(a0)
restart2
        bset.b  d0,(a0)         ; see if busy
        nop
        bne             restart2

        move.l  d1,Src_Addr(a1)
        move.b  #8,Skew(a1)
        suba    #8,a2
        move.l  a2,Dst_Addr(a1)
        move.w  d2,Y_Count(a1)
        move.b  #mLineBusy+mLineSmudge,(a0)
restart3
        bset.b  d0,(a0)         ; see if busy
        nop
        bne             restart3


        move.l  d1,Src_Addr(a1)
        move.b  #12,Skew(a1)
        suba    #8,a2
```

continued

```
        move.l  a2,Dst_Addr(a1)
        move.w  d2,Y_Count(a1)
        move.b  #mLineBusy+mLineSmudge,(a0)
restart4
        bset.b  d0,(a0)         ; see if busy
        nop
        bne             restart4
        rts


rpl_line
        move.l  a2,Src_Addr(a1)
        adda.w  #160,a2
        move.l  a2,Dst_Addr(a1)
        move.w  d2,Y_Count(a1)
        move.b  #mLineBusy,(a0)
restart5
        bset.b  d0,(a0)         ; see if busy
        nop
        bne             restart5
        rts

        public _copy_screen
        ;copy_screen(s, d)
        ;       copy 32K screen pretty fast, but not with blitter
_copy_screen
        move.l 4(sp),a0
        move.l 8(sp),a1
        move.w #999,d0  ;32bytes/loop x 1000-1
cslp
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        move.l  (a0)+,(a1)+
        dbra    d0,cslp
        rts
```

X386B1.C Accompanies SCO XENIX 386 by Edwin J. Lau, BYTE, December, 1987, page 190

```
/* This is Xenix benchmark program 1 */

#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/times.h>

main(argc, argv)
int argc;
char *argv[];
{
    extern int optind;
    extern char *optarg;
    int c, errflag, fd, wnum, i, id, size;
    char *mptr;
    struct tms tp;
    long et1, et2;

    size = 0;
    wnum = 50;
    errflag = 0;
    while ((c = getopt(argc, argv, "n:")) != EOF)
        switch(c) {
        case 'n':
            wnum = atoi(optarg);
#ifdef DEBUG
            printf("write %d times\n", wnum);
#endif
            break;

       case '?':
            errflag = 1;
            break;
       )
```

```
    if (errflag) {
        printf("Usage: xb1 (getpid) [ -n count ] \n");
        exit(1);
    }

    et1 = times(&tp);
    for (i = 0; i < wnum; i++) {
        id = getpid();
    }
    et2 = times(&tp);
    printf("elapse time getpid (sec) = %f\n", (float) (et2 - et1)/50.);
}
```

X386B2.C Accompanies SCO XENIX 386 by Edwin J. Lau, BYTE, December, 1987, page 190

```
/* This is Xenix benchmark program 2 */
#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/times.h>

char bss[32768];

main(argc, argv)
int argc;
char *argv[];
{
    extern int optind;
    extern char *optarg;
    int c, errflag, fnum, i, id;
    struct tms tp;
    long et1, et2;
    char *mptr;

    fnum = 50;
    errflag = 0;
    while ((c = getopt(argc, argv, "n:f:")) != EOF)
        switch(c) {
        case 'n':
            fnum = atoi(optarg);
#ifdef DEBUG
            printf("fork %d times\n", fnum);
#endif
            break;

        case '?':
            errflag = 1;
            break;
        }

    if (errflag) {
        printf("Usage: xb2 (fork) [ -n count ] \n");
        exit(1);
    }

    et1 = times(&tp);
    for (i = 0; i < fnum; i++) {
        if ((id = fork()) == 0) {
            /* This is the child process */
            exit(0);
        }
    }
    et2 = times(&tp);
    printf("fork + 32k heap elapse time (sec) = %f\n",
        (float) (et2 - et1)/50.);
}
```

X386B3.C Accompanies SCO XENIX 386 by Edwin J. Lau, BYTE, December, 1987, page 190

```
/* This is Xenix benchmark program 3 */
```

continued

```
#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/times.h>

main(argc, argv)
int argc;
char *argv[];
{
    extern int optind;
    extern char *optarg;
    int c, errflag, fd, wnum, i, id, size;
    char *mptr;
    struct tms tp;
    long et1, et2;

    size = 0;
    wnum = 50;
    errflag = 0;
    while ((c = getopt(argc, argv, "n:s:")) != EOF)
        switch(c) {
        case 'n':
            wnum = atoi(optarg);
#ifdef DEBUG
            printf("write %d times\n", wnum);
#endif
            break;

        case 's':
            size = atoi(optarg);
            printf("size = %d \n", size);
#ifdef DEBUG
            printf("size = %d \n", size);
#endif
            break;

        case '?':
            errflag = 1;
            break;
        }

    if (errflag || (size == 0)) {
        printf("Usage: xb3 (write) [ -n count ] -s size \n");
        exit(1);
    }

    if ((fd = creat("junk", 0644)) == -1) {
        printf("error: can't create file\n");
        exit(1);
    }
    mptr = malloc(size);
    for (i = 0; i < size; i++) mptr[i] = 'a';
    et1 = times(&tp);
    for (i = 0; i < wnum; i++) {
        write(fd, mptr, size);
    }
    et2 = times(&tp);
    printf("elapse time write (sec) = %f\n", (float) (et2 - et1)/50.);
}
```

‹386B4.C Accompanies SCO XENIX 386 by Edwin J. Lau, BYTE, December, 1987, page 190

```
/* This is Xenix benchmark program 4 */
#include <stdio.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/times.h>

main(argc, argv)
int argc;
char *argv[];
{
    extern int optind;
    extern char *optarg;
    int c, errflag, fd, wnum, i, id, size;
    char *mptr;
    struct tms tp;
    long et1, et2;
```

```
    size = 0;
    wnum = 50;
    errflag = 0;
    while ((c = getopt(argc, argv, "n:s:")) != EOF)
        switch(c) {
        case 'n':
            wnum = atoi(optarg);
#ifdef DEBUG
            printf("write %d times\n", wnum);
#endif
            break;

        case 's':
            size = atoi(optarg);
#ifdef DEBUG
            printf("size = %d \n", size);
#endif
            break;

        case '?':
            errflag = 1;
            break;
        }

    if (errflag || (size == 0)) {
        printf("Usage: xb4 (read) [ -n count ] -s size \n");
        exit(1);
    }

    if ((fd = open("junk", 0644)) == -1) {
        printf("error: can't open file\n");
        exit(1);
    }
    mptr = malloc(size);
    for (i = 0; i < size; i++) mptr[i] = 'a';

    et1 = times(&tp);
    for (i = 0; i < wnum; i++) {
        read(fd, mptr, size);
    }
    et2 = times(&tp);
    printf("elapse time read (sec) = %f\n", (float) (et2 - et1)/50.);
}
```

X386B5.C Accompanies SCO XENIX 386 by Edwin J. Lau, BYTE, December, 1987, page 190

```
/* This is Xenix benchmark program 5 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/times.h>

main(argc, argv)
int argc;
char *argv[];
{
    extern int optind;
    extern char *optarg;
    int c, errflag, fd, wnum, i, id, size;
    char *mptr;
    struct tms tp;
    long et1, et2;

    wnum = 50;
    errflag = 0;
    while ((c = getopt(argc, argv, "n:s:")) != EOF)
        switch(c) {
        case 'n':
            wnum = atoi(optarg);
#ifdef DEBUG
            printf("write %d times\n", wnum);
#endif
            break;

        case '?':
            errflag = 1;
            break;
        }
```

continued

```
    if (wnum == 0) {
        printf("Usage: xb5 (screen1) -n size \n");
        exit(1);
    }

    et1 = times(&tp);
    for (i = 0; i < wnum; i++) {
        printf("x");
    }
    et2 = times(&tp);
    printf("elapse time write (sec) = %f\n", (float) (et2 - et1)/50.);
}
```

```
:386B6.C Accompanies SCO XENIX 386 by Edwin J. Lau, BYTE, December,
:987, page 190
```

```
/* This is Xenix benchmark program 6 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/times.h>

main(argc, argv)
int argc;
char *argv[];
{
    extern int optind;
    extern char *optarg;
    int c, errflag, fd, wnum, i, id, size;
    char *mptr;
    struct tms tp;
    long et1, et2;

    wnum = 50;
    errflag = 0;
    while ((c = getopt(argc, argv, "n:s:")) != EOF)
        switch(c) {
        case 'n':
            wnum = atoi(optarg);
#ifdef DEBUG
            printf("write %d times\n", wnum);
#endif
            break;

        case '?':
            errflag = 1;
            break;
        }

    if (errflag ) {
        printf("Usage: xb6 (screen2) [ -n count ] \n");
        exit(1);
    }

    et1 = times(&tp);
    for (i = 0; i < wnum; i++) {
        printf("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
");
    }
    et2 = times(&tp);
    printf("elapse time write (sec) = %f\n", (float) (et2 - et1)/50.);
}
```

```
FERRET.ASM accompanies "Ferret: An Image Processor" by Clifford Harris,
BYTE, December, 1987, page 317.
```

```
*                   PROGRAM FERRET
*                   THIS IS THE PROGRAM FOR IMAGE PROCESSING
*                   RCA SID-504 CCD IMAGE
*                   by
*                   Clifford Harris
*                   99 Mason Rd.
*                   Yerington, Nv. 89447
*
*                   This version does not include the telescope controller.
```

```
ROWS      = 403
COLUMNS   = 256          * MUST BE MULTIPLE OF 16
BASEROW   = 70
BASECOL   = 0
TOPROW    = BASEROW+403
BASECOL1  = 64
BASECOL2  = 336
A_D_DATA  = $FF0000      *A/D CONVERTER
IOBASE    = $FF0000      *BASE ADDRESS I/O BLOCK
SIO       = IOBASE+$70   *BASE ADDRESS INTERFACER 4 BOARD
VECOUT    = SIO          *INTERFACER 4 DATA REGISTER
VECSTAT   = SIO+1        *I 4 STATUS REG
MODE      = SIO+2        *I 4 MODE REG
COMMAND   = SIO+3        *I 4 COMMAND REG
SELECT    = SIO+7        *I 4 SELECT REG
PARDATA   = SIO+2        *I 4 PARALLEL DATA REG
SYSBASE   = IOBASE+$50   *BASE ADDRESS SYSTEM SUPPORT 1
SYSIO     = SYSBASE+$C   *SYSTEM SUPPORT CONSOLE DATA REGISTER
SYSST     = SYSBASE+$D   *SYSTEM SUPPORT CONSOLE STATUS REG
MATHDATA  = SYSBASE+8    *MATH CHIP DATA REGISTER
MATHCMD   = SYSBASE+9    *MATH CHIP COMMAND REG
CLKCMD    = SYSBASE+$A
CLKDATA   = SYSBASE+$B

          .TEXT
*********
*         TOP MENU
PGM:
 JSR      VECINIT
 MOVE.L   #$FFFFFFFF,BADMAP
 MOVE.W   #170,FRAMEY     * X,Y STARTS FRAME NEAR MIDDLE
 MOVE.W   #400,FRAMEX
 MOVE.W   #1,EXPOSURE
 MOVE.B   #$30,EXPTIME1
 MOVE.B   #$30,EXPTIME2
 MOVE.B   #$30,EXPTIME3
 MOVE.B   #$31,EXPTIME4
 MOVE.B   #2,HORTIME
 MOVE.B   #56,VERTIME
 MOVE.B   #40,ADTIME
 JSR      TEST
 MOVE.B   #'Q',VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #1,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #1,VECOUT
 MOVE.W   #256,D0
 MOVE.B   #255,RED
 MOVE.B   #255,GREEN
 MOVE.B   #255,BLUE
HICOLOR:
 JSR      COLOUT
 SUB.W    #1,D0
 BNE      HICOLOR
 MOVE.L   #LOGDATA,A0
 MOVE.L   #0,D7
LOGLOAD:
 MOVE.L   #1133,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.L   D7,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$08,MATHCMD
 MOVE.B   #$12,MATHCMD
 MOVE.B   #$1E,MATHCMD
 JSR      YANKD0
 MOVE.W   D0,(A0)+
 ADD.L    #1,D7
 CMP.L    #4096,D7
 BNE      LOGLOAD
 MOVE.W   #400,CUSVALL
 MOVE.W   #528,CUSVALR
 MOVE.W   #464,CUSVALH
 MOVE.W   #271,CUSVALV
 MOVE.W   #464,CUSVH
 MOVE.W   #271,CUSVV
```

continued

```
 MOVE.L    #FCB1,A0
 ADD.L     #8,A0
 MOVE.B    #'0',(A0)
 ADD.L     #4,A0
 MOVE.B    #24,D0
FCBZ0:
 MOVE.B    #0,(A0)+
 SUB.B     #1,D0
 BNE       FCBZ0
* JSR       RESTORE
 MOVE.L    #PAGE1,A1
 MOVE.L    #PAGE3,A4
 MOVE.L    #$19300,D0
DRKLOAD:
 MOVE.W    (A1)+,(A4)+
 SUB.L     #1,D0
 BNE       DRKLOAD
 MOVE.L    #FCB1,A0
 ADD.L     #8,A0
 MOVE.B    #'1',(A0)
 ADD.L     #4,A0
 MOVE.B    #24,D0
FCBZ1:
 MOVE.B    #0,(A0)+
 SUB.B     #1,D0
 BNE       FCBZ1
* JSR       RESTORE
 MOVE.L    #PAGE1,A1
 MOVE.L    #PAGE4,A4
 MOVE.L    #$19300,D0
FLTLOAD:
 MOVE.W    (A1)+,(A4)+
 SUB.L     #1,D0
 BNE       FLTLOAD

TOP:
 JSR       SCREEN
 JSR       TESTIN
 CLR.L     D0
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 CMP.B     #'0',D0
 BNE       PG1
 JSR       AUTOMODE
PG1:
 CMP.B     #'1',D0
 BNE       PG2
 JSR       MANMODE
PG2:
 CMP.B     #'2',D0
 BNE       PG3
 JSR       TIMING
PG3:
 CMP.B     #'3',D0
 BNE       PG4
 JSR       EXPCON
PG4:
 CMP.B     #'Q',D0
 BEQ       ALLDONE
 BRA       TOP
ALLDONE:
 MOVE.L    #ENDMSG,A1
 JSR       MSGOUT
 RTS    *TO CP/M
*********
*          SUBROUTINE EXPCON FOR SETTING EXPOSURE TIME
EXPCON:
 MOVEM.L   D0-D4/A0,-(A7)
TRYAGAIN:
 MOVE.B    #'0',EXPTIME1
 MOVE.B    #'0',EXPTIME2
 MOVE.B    #'0',EXPTIME3
 MOVE.B    #'1',EXPTIME4
 MOVE.L    #EXPMSG,A1
 JSR       MSGOUT
 JSR       INNOTIME
 CLR       D0
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 CMP.B     #'0',D0
 BLT       TRYAGAIN
 CMP.B     #'9',D0
```

```
 BGT       TRYAGAIN
 MOVE.B    D0,SYSIO
 MOVE.B    D0,EXPTIME1
 JSR       INNOTIME
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 CMP.B     #'0',D0
 BLT       TRYAGAIN
 CMP.B     #'9',D0
 BGT       TRYAGAIN
 MOVE.B    D0,SYSIO
 MOVE.B    D0,EXPTIME2
 JSR       INNOTIME
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 CMP.B     #'0',D0
 BLT       TRYAGAIN
 CMP.B     #'9',D0
 BGT       TRYAGAIN
 MOVE.B    D0,SYSIO
 MOVE.B    D0,EXPTIME3
 JSR       INNOTIME
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 CMP.B     #'0',D0
 BLT       TRYAGAIN
 CMP.B     #'9',D0
 BGT       TRYAGAIN
 MOVE.B    D0,SYSIO
 MOVE.B    D0,EXPTIME4
EXPOUT:
 CLR.L     D1
 MOVE.B    EXPTIME4,D1
 SUB.B     #$30,D1
 CLR.L     D0
 MOVE.B    EXPTIME3,D0
 SUB.B     #$30,D0
 MULU      #10,D0
 ADD.W     D0,D1
 CLR.L     D0
 MOVE.B    EXPTIME2,D0
 SUB.B     #$30,D0
 MULU      #100,D0
 ADD.W     D0,D1
 CLR.L     D0
 MOVE.B    EXPTIME1,D0
 SUB.B     #$30,D0
 MULU      #1000,D0
 ADD.W     D0,D1
 CMP.W     #0,D1
 BNE       EXPDONE
 MOVE.W    #1,D1
EXPDONE:
 MOVE.W    D1,EXPOSURE
 MOVEM.L   (A7)+,D0-D4/A0
 RTS
*********
*          SUBROUTINE TIMING FOR SETTING CCD PULSES
TIMING:
 MOVEM.L   D0-D4/A0,-(A7)
 JSR       TESTIN
 CLR.L     D0
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 SUB.B     #$30,D0
 LSL.B     #1,D0
 MOVE.B    D0,HORTIME
 JSR       TESTIN
 CLR.L     D0
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 SUB.B     #$30,D0
 LSL.B     #3,D0
 MOVE.B    D0,VERTIME
 JSR       TESTIN
 CLR.L     D0
 MOVE.B    SYSIO,D0
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 SUB.B     #$30,D0
 LSL.B     #3,D0
```

continued

```
 MOVE.B   D0,ADTIME
 MOVEM.L  (A7)+,D0-D4/A0
 RTS
*********
MANMODE:
 MOVEM.L  D0-D7/A0-A6,-(A7)
 JSR      MANSCR
MCOMAND:
 MOVE.L   #RDYMSG2,A1
 JSR      ACKMSG
 JSR      TESTIN
 CLR.L    D0
 MOVE.B   SYSIO,D0
 CMP.B    #'0',D0
 BNE      MJ1
 MOVE.L   #ACKMSG20,A1
 JSR      ACKMSG
 JSR      RCAIN
MJ1:
 CMP.B    #'1',D0
 BNE      MJ2
 MOVE.L   #ACKMSG21,A1
 JSR      ACKMSG
 JSR      PTOS
MJ2:
 CMP.B    #'2',D0
 BNE      MJ3
 MOVE.L   #ACKMSG22,A1
 JSR      ACKMSG
 JSR      AVERAGE
MJ3:
 CMP.B    #'3',D0
 BNE      MJ4
 MOVE.L   #ACKMSG23,A1
 JSR      ACKMSG
 JSR      PIXELFIX
MJ4:
 CMP.B    #'4',D0
 BNE      MJ5
 MOVE.L   #ACKMSG24,A1
 JSR      ACKMSG
 JSR      COLOR
MJ5:
 CMP.B    #'5',D0
 BNE      MJ6
 MOVE.L   #ACKMSG25,A1
 JSR      ACKMSG
 JSR      BIGSHOW
MJ6:
 CMP.B    #'6',D0
 BNE      MJ7
 MOVE.L   #ACKMSG26,A1
 JSR      ACKMSG
 JSR      LOG
MJ7:
 CMP.B    #'7',D0
 BNE      SJ1
 MOVE.L   #ACKMSG27,A1
 JSR      ACKMSG
 MOVE.B   #1,PAGE
 MOVE.W   #BASECOL1,X
 JSR      SHOW
SJ1:
 CMP.B    #'8',D0
 BNE      SJ2
 MOVE.L   #ACKMSG28,A1
 JSR      ACKMSG
 MOVE.B   #2,PAGE
 MOVE.W   #BASECOL2,X
 JSR      SHOW
SJ2:
 CMP.B    #'9',D0
 BNE      MJ8
 MOVE.L   #ACKMSG29,A1
 JSR      ACKMSG
 JSR      GROUPS
MJ8:
 CMP.B    #'A',D0
 BNE      MJ9
 MOVE.L   #ACKMSG2A,A1
 JSR      ACKMSG
 JSR      MAGNIFY
MJ9:
 CMP.B    #'B',D0
 BNE      MJ10
 MOVE.L   #ACKMSG2B,A1
```

```
  JSR       ACKMSG
  JSR       TESTOUT
  MOVE.B    $1A,SYSIO
  JSR       REPORT
  JSR       MANSCR
MJ10:
  CMP.B     #'C',D0
  BNE       MJ11
  MOVE.L    #ACKMSG3C,A1
  JSR       ACKMSG
  JSR       FILEPICK
  JSR       SAVE
MJ11:
  CMP.B     #'D',D0
  BNE       SJ3
  MOVE.L    #ACKMSG2D,A1
  JSR       ACKMSG
  JSR       SCRNVAL
SJ3:
  CMP.B     #'E',D0
  BNE       SJ4
  MOVE.L    #ACKMSG2E,A1
  JSR       ACKMSG
  JSR       WHIRL
SJ4:
  CMP.B     #'F',D0
  BNE       MJ12
  MOVE.L    #ACKMSG3F,A1
  JSR       ACKMSG
  JSR       FILEPICK
  JSR       RESTORE
MJ12:
  CMP.B     #'G',D0
  BNE       MJ13
  MOVE.L    #ACKMSG2G,A1
  JSR       ACKMSG
  JSR       DRKFIELD
MJ13:
  CMP.B     #'H',D0
  BNE       MJ14
  MOVE.L    #ACKMSG2H,A1
  JSR       ACKMSG
  JSR       SUBSTOP
MJ14:
  CMP.B     #'I',D0
  BNE       MJ15
  MOVE.L    #ACKMSG2I,A1
  JSR       ACKMSG
  JSR       FLTFIELD
MJ15:
  CMP.B     #'J',D0
  BNE       MJ16
  MOVE.L    #ACKMSG2J,A1
  JSR       ACKMSG
  JSR       BIAS
MJ16:
  CMP.B     #'K',D0
  BNE       MJ17
  MOVE.L    #ACKMSG2K,A1
  JSR       ACKMSG
  JSR       HISTCURS
MJ17:
  CMP.B     #'L',D0
  BNE       MJ18
  MOVE.L    #ACKMSG2L,A1
  JSR       ACKMSG
  JSR       SCRNCUR
MJ18:
  CMP.B     #'M',D0
  BNE       MJ19
  MOVE.L    #ACKMSG2M,A1
  ASL.L     #1,D0
  ADD.L     D0,A1
  CLR.L     D0
  MOVE.W    (A1),D0
  JSR       DIGITS4
  RTS
CDONE:
  MOVE.W    CUSVALH,D0
  MOVE.W    CUSVH,D3
  MOVE.W    D3,CUSVALH
  MOVE.W    D0,CUSVH
  MOVE.W    D3,D1
  MOVE.W    CUSVALV,D0
```

continued

```
 MOVE.W   CUSVV,D3
 MOVE.W   D3,CUSVALV
 MOVE.W   D0,CUSVV
 MOVE.W   D3,D2
 JSR      CB
 JSR      TEST
 MOVE.B   #'B',VECOUT     *ENABLE ALL BIT PLANES
 JSR      TEST
 MOVE.B   #$FF,VECOUT
 JSR      TEST
 MOVE.B   #$1,VECOUT
 MOVEM.L  (A7)+,D0-D7/A0-A1
 RTS
*********
VECMSG:
 MOVE.L   D0,-(A7)
 MOVE.W   #255,D0
 JSR      C
 JSR      TEST
 MOVE.B   #'R',VECOUT
 JSR      TEST
 MOVE.B   #'A',VECOUT
 MOVE.W   #600,X
 MOVE.W   #110,Y
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'X',VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 MOVE.W   #600,X
 MOVE.W   #70,Y
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'Y',VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 MOVE.W   #600,X
 MOVE.W   #220,Y
 JSR      TEST
 MOVE.B   #'J',VECOUT
 JSR      TEST
 MOVE.B   #'M',VECOUT
 JSR      TEST
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$17,MATHCMD
 MOVE.B   #$12,MATHCMD
 CLR.L    D0
 CLR.L    D1
 MOVE.W   CUSVALH,D0
 MOVE.W   CUSVH,D1
 CMP.W    D1,D0
 BGT      LSE2
 EXG      D1,D0
LSE2:
 SUB.L    D1,D0
 MULU     #39,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$17,MATHCMD
 MOVE.B   #$12,MATHCMD
 MOVE.B   #$10,MATHCMD
 MOVE.B   #$01,MATHCMD
 MOVE.L   #5,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$10,MATHCMD
 MOVE.L   #10,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$13,MATHCMD
 MOVE.B   #$1E,MATHCMD
 MOVEM.L  (A7)+,D0-D2
 MOVE.W   #255,D0
 JSR      C
 MOVE.W   #600,X
 MOVE.W   #190,Y
 CLR.L    D0
 JSR      YANKD0
 AND.L    #$0000FFFF,D0
 JSR      DIGITS4
 MOVE.W   #600,X
 MOVE.W   #90,Y
```

```
 CLR.L    D0
 MOVE.W   D1,D0
 SUB.W    #335,D0
 JSR      DIGITS4
 MOVE.W   #600,X
 MOVE.W   #50,Y
 CLR.L    D0
 MOVE.W   D2,D0
 SUB.W    #69,D0
 JSR      DIGITS4
 MOVE.W   #600,X
 MOVE.W   #390,Y
 MOVE.L   #PAGE2,A1
 CLR.L    D0
 MOVE.W   CUSVALV,D0
 SUB.L    #70,D0
 MULU     #256,D0
 CLR.L    D4
 MOVE.W   CUSVALH,D4
 ADD.L    D4,D0
 SUB.L    #336,D0
 BRA      KEYBDIN
CW:
 MOVE.W   #511,D0
 JSR      C
 BRA      CVAL
CB:
 MOVE.W   #0,D0
 JSR      C
CVAL:
 SUB.W    #4,D1
 MOVE.W   D1,X
 MOVE.W   D2,Y
 JSR      M
 ADD.W    #8,D1
 MOVE.W   D1,X
 MOVE.W   D2,Y
 JSR      L
 SUB.W    #4,D1
 SUB.W    #4,D2
 MOVE.W   D1,X
 MOVE.W   D2,Y
 JSR      M
 ADD.W    #8,D2
 MOVE.W   D1,X
 MOVE.W   D2,Y
 JSR      L
 SUB.W    #4,D2
 MOVE.W   D1,D7
 SUB.W    #272,D7
 SUB.W    #4,D7
 MOVE.W   D7,X
 MOVE.W   D2,Y
 JSR      M
 ADD.W    #8,D7
 MOVE.W   D7,X
 MOVE.W   D2,Y
 JSR      L
 SUB.W    #4,D7
 SUB.W    #4,D2
 MOVE.W   D7,X
 MOVE.W   D2,Y
 JSR      M
 ADD.W    #8,D2
 MOVE.W   D7,X
 MOVE.W   D2,Y
 JSR      L
 SUB.W    #4,D2
LETSSEE:
 MOVEM.L  D0-D2,-(A7)
 CLR.L    D0
 CLR.L    D1
 MOVE.W   CUSVALV,D0
 MOVE.W   CUSVV,D1
 CMP.W    D1,D0
 BGT      LSE1
 EXG      D1,D0
LSE1:
 SUB.L    D1,D0
 MULU     #31,D0
 JSR      STUFFD0
 JSR      CB
 MOVE.B   SYSIO,D0
```

continued

```
 CMP.B    #$08,D0        *LEFT ARROW ?
 BEQ      CLEFT
 CMP.B    #$0C,D0        *RIGHT ARROW ?
 BEQ      CRIGHT
 CMP.B    #$0B,D0        *LEFT ARROW ?
 BEQ      CUP
 CMP.B    #$16,D0        *RIGHT ARROW ?
 BEQ      CDOWN
 CMP.B    #$0D,D0        *CARRIGE RETURN ?
 BEQ      CDONE
 CMP.B    #'L',D0
 BNE      OTHER
 JSR      CW
 JSR      GRAPH
 BRA      KEYBDIN
OTHER:
 CMP.B    #' ',D0
 BNE      KEYBDIN
 JSR      CW
 MOVE.W   CUSVALH,D0
 MOVE.W   CUSVH,D3
 MOVE.W   D3,CUSVALH
 MOVE.W   D0,CUSVH
 MOVE.W   D3,D1
 MOVE.W   CUSVALV,D0
 MOVE.W   CUSVV,D3
 MOVE.W   D3,CUSVALV
 MOVE.W   D0,CUSVV
 MOVE.W   D3,D2
 BRA      KEYBDIN

CRIGHT:
 CMP.W    #591,D1
 BGT      KEYBDIN
 ADD.W    #1,D1
 MOVE.W   D1,CUSVALH
 JSR      LETSSEE
 BRA      KEYBDIN
CLEFT:
 CMP.W    #337,D1
 BLT      KEYBDIN
 SUB.W    #1,D1
 MOVE.W   D1,CUSVALH
 JSR      LETSSEE
 BRA      KEYBDIN
CUP:
 CMP.W    #472,D2
 BGT      KEYBDIN
 ADD.W    #1,D2
 MOVE.W   D2,CUSVALV
 JSR      LETSSEE
 BRA      KEYBDIN
CDOWN:
 CMP.W    #71,D2
 BLT      KEYBDIN
 SUB.W    #1,D2
 MOVE.W   D2,CUSVALV
 JSR      LETSSEE
 CMP.W    D1,D2
 BGT      NOSWITCH
 EXG      D1,D2
NOSWITCH:
 SUB.W    D1,D2
 CMP.W    #200,D2
 BGT      ENUF
 MOVE.W   #0,(A1)+
 SUB.L    #1,D0
 BNE      DIFFER1
 BRA      DIFFOUT
ENUF:
 SUB.L    #2,A2
 MOVE.W   (A2)+,D2
 MOVE.W   D2,(A1)+
 SUB.L    #1,D0
 BNE      DIFFER1
DIFFOUT:
 MOVEM.L  (A7)+,D0-D3/A0-A3
 RTS
*********
SUBCONST:
 MOVEM.L  D0-D3/A0-A2,-(A7)
 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A2
 MOVE.L   #$19300,D0
 MOVE.W   CUSVALL,D3
```

```
 SUB.W    #337,D3
 ASL.W    #4,D3
SUBC1:
 MOVE.W   (A2)+,D1
 SUB.W    D3,D1
 CMP.W    #0,D1
 BGT      SUBC2
 MOVE.W   #0,D1
SUBC2:
 MOVE.W   D1,(A1)+
 SUB.L    #1,D0
 BNE      SUBC1
 MOVEM.L  (A7)+,D0-D3/A0-A2
 RTS
*********
*         SCRNCUR
SCRNCUR:
 MOVEM.L  D0-D7/A0-A1,-(A7)
 MOVE.W   CUSVALH,D1
 MOVE.W   CUSVALV,D2
 JSR      TEST
 MOVE.B   #'B',VECOUT    *ENABLE ONLY 9th BIT PLANE
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #1,VECOUT
 JSR      VECMSG
 JSR      LETSSEE
KEYBDIN:
 JSR      CW
KEYWAIT:
 BTST     #1,SYSST
 BEQ      KEYWAIT
 MULU     D3,D2
 ADD.L    D2,D4
 MOVE.W   D3,(A1)+
 SUB.L    #1,D0
 BNE      DELTAP2
 DIVU     #7936,D4
 AND.L    #$0000FFFF,D4
 DIVU     #13,D4
 AND.L    #$0000FFFF,D4
 MOVE.L   D4,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$01,MATHCMD
 MOVE.B   #$1E,MATHCMD
 JSR      YANKD0
 MOVE.L   #PAGE1,A1
 MOVE.L   #2560,D3
MEANRES:
 MOVE.W   D1,(A1)+
 SUB.L    #1,D3
 BNE      MEANRES
 MOVE.L   #2560,D3
DEVRES:
 MOVE.W   D0,(A1)+
 SUB.L    #1,D3
 BNE      DEVRES
 MOVEM.L  (A7)+,D0-D4/A0-A2
 RTS
*********
ADCONST:
 MOVEM.L  D0-D3/A0-A2,-(A7)
 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A2
 MOVE.L   #$19300,D0
 MOVE.W   CUSVALL,D3
 SUB.W    #337,D3
 ASL.W    #4,D3
ADDC1:
 MOVE.W   (A2)+,D1
 ADD.W    D3,D1
 CMP.W    #4095,D1
 BLT      ADDC2
 MOVE.W   #4095,D1
ADDC2:
 MOVE.W   D1,(A1)+
 SUB.L    #1,D0
 BNE      ADDC1
 MOVEM.L  (A7)+,D0-D3/A0-A2
 RTS
*********
```

continued

```
DIFFER:
 MOVEM.L  D0-D3/A0-A3,-(A7)
 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A2
 MOVE.L   #$19300,D0
DIFFER1:
 CLR.L    D1
 CLR.L    D2
 MOVE.W   (A2)+,D2
 MOVE.W   (A1),D1
 JSR      RCAIN
 JSR      MAKEDARK
 MOVE.L   #1,D1
 MOVE.L   #2,D2
AUTOIN:
 JSR      RCAIN
 MOVE.L   #PAGE3,A3
 MOVE.L   #PAGE1,A1
 MOVE.L   #$19300,D0
AUTOLOP1:
 CLR.L    D3
 CLR.L    D4
 MOVE.W   (A3),D3
 MULU     D1,D3
 MOVE.W   (A1),D4
 ADD.L    D4,D3
 DIVU     D2,D3
 MOVE.W   D3,(A3)+
 MOVE.W   D3,(A1)+
 SUB.L    #1,D0
 BNE      AUTOLOP1
 ADD.L    #1,D1
 ADD.L    #1,D2
 CMP.L    #10,D1
 BNE      AUTOIN
AUTOOUT:
 MOVEM.L  (A7)+,D0-D7/A0-A6
 RTS
*********
STATS:
 MOVEM.L  D0-D4/A0-A2,-(A7)
 MOVE.L   #PAGE2,A2
 MOVE.L   #$19300,D0
 CLR.L    D1
 CLR.L    D2
SUMP2:
 MOVE.W   (A2)+,D2
 ADD.L    D2,D1
 SUB.L    #1,D0
 BNE      SUMP2
 DIVU     #7936,D1
 AND.L    #$0000FFFF,D1
 DIVU     #13,D1
 AND.L    #$0000FFFF,D1
 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A2
 MOVE.L   #$19300,D0
 CLR.L    D2
 CLR.L    D3
 CLR.L    D4
DELTAP2:
 CLR.L    D2
 MOVE.W   (A2)+,D2
 MOVE.W   D1,D3
 CMP      D3,D2
 BGT      GOON
 EXG      D3,D2
GOON:
 SUB.W    D3,D2
 MOVE.W   D2,D3
 JSR      ACKMSG
* JSR      BLACKOUT
MJ19:
 CMP.B    #'N',D0
 BNE      MJ20
 JSR      NEGATIVE
MJ20:
 CMP.B    #'O',D0
 BNE      MJ21
 JSR      ADCONST
MJ21:
 CMP.B    #'P',D0
 BNE      MJ22
 JSR      SUBCONST
```

```
MJ22:
 CMP.B    #'R',D0
 BNE      MJ23
 JSR      MAKEDARK
MJ23:
 CMP.B    #'S',D0
 BNE      MJ24
 JSR      MAKEFLAT
MJ24:
 CMP.B    #'T',D0
 BNE      MJ25
 JSR      STAROUT5
MJ25:
 CMP.B    #'U',D0
 BNE      MJ26
 JSR      THREED
MJ26:
 CMP.B    #'V',D0
 BNE      MJ27
 JSR      HIGHPASS
MJ27:
 CMP.B    #'W',D0
 BNE      MJ28
 JSR      MAPMAKER
MJ28:
 CMP.B    #'X',D0
 BNE      MJ29
 JSR      STAROUT
MJ29:
 CMP.B    #'Y',D0
 BNE      MJ30
 JSR      STATS
MJ30:
 CMP.B    #'Z',D0
 BNE      MJ36
 JSR      DIFFER
MJ36:
 CMP.B    #'Q',D0
 BEQ      MANOUT
 BRA      MCOMAND
MANOUT:
 MOVEM.L  (A7)+,D0-D7/A0-A6
 RTS    *TO TOP
*********
AUTOMODE:
 MOVEM.L  D0-D7/A0-A6,-(A7)
 MOVE.B   #1,VECOUT
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'S',VECOUT
 JSR      TEST
 MOVE.B   #'e',VECOUT
 JSR      TEST
 MOVE.B   #'p',VECOUT
 JSR      TEST
 MOVE.B   #'a',VECOUT
 JSR      TEST
 MOVE.B   #'r',VECOUT
 JSR      TEST
 MOVE.B   #'a',VECOUT
 JSR      TEST
 MOVE.B   #'t',VECOUT
 JSR      TEST
 MOVE.B   #'i',VECOUT
 JSR      TEST
 MOVE.B   #'o',VECOUT
 JSR      TEST
 MOVE.B   #'n',VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 MOVE.W   #600,X
 MOVE.W   #210,Y
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'i',VECOUT
 JSR      TEST
 MOVE.B   #'n',VECOUT
 JSR      TEST
 MOVE.B   #' ',VECOUT
 JSR      TEST
```

continued

```
 MOVE.B   #'a',VECOUT
 JSR      TEST
 MOVE.B   #'r',VECOUT
 JSR      TEST
 MOVE.B   #'c',VECOUT
 JSR      TEST
 MOVE.B   #' ',VECOUT
 JSR      TEST
 MOVE.B   #'
            '",VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 MOVE.W   #600,X
 MOVE.W   #430,Y
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'1',VECOUT
 JSR      TEST
 MOVE.B   #'2',VECOUT
 JSR      TEST
 MOVE.B   #' ',VECOUT
 JSR      TEST
 MOVE.B   #'B',VECOUT
 JSR      TEST
 MOVE.B   #'i',VECOUT
 JSR      TEST
 MOVE.B   #'t',VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 MOVE.W   #600,X
 MOVE.W   #420,Y
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'P',VECOUT
 JSR      TEST
 MOVE.B   #'i',VECOUT
 JSR      TEST
 MOVE.B   #'x',VECOUT
 JSR      TEST
 MOVE.B   #'e',VECOUT
 JSR      TEST
 MOVE.B   #'l',VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 MOVE.W   #600,X
 MOVE.W   #410,Y
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 JSR      TEST
 MOVE.B   #'V',VECOUT
 JSR      TEST
 MOVE.B   #'a',VECOUT
 JSR      TEST
 MOVE.B   #'l',VECOUT
 JSR      TEST
 MOVE.B   #'u',VECOUT
 JSR      TEST
 MOVE.B   #'e',VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 JSR      TEST
 MOVE.B   #'R',VECOUT
 JSR      TEST
 MOVE.B   #'E',VECOUT
 JSR      TEST
 MOVE.B   #'J',VECOUT
 JSR      TEST
 MOVE.B   #'M',VECOUT
 JSR      TEST
 MOVE.B   #2,VECOUT
 MOVE.L   (A7)+,D0
 RTS
*********
*        SUBROUTINE FOURDIG
*        WRITES VALUE OF D0 TO VECTRIX SCREEN IN UP TO FOUR DECIMAL DIGITS
DIGITS4:
 MOVEM.L  D0-D1/A0,-(A7)
 MOVE.L   D0,D1
 MOVE.W   #511,D0
 JSR      C
 JSR      TEST
```

```
 MOVE.B   #'R',VECOUT
 JSR      TEST
 MOVE.B   #'A',VECOUT
 JSR      M
 JSR      TEST
 MOVE.B   #'$',VECOUT
 MOVE.L   D1,D0
 JSR      STASHD0
 JSR      KRUNCH
 MOVE.L   #DECIMAL,A0
 ADD.L    #6,A0
 JSR      TEST
 MOVE.B   (A0)+,VECOUT
 JSR      TEST
 MOVE.B   (A0)+,VECOUT
 JSR      TEST
 MOVE.B   (A0)+,VECOUT
 JSR      TEST
 MOVE.B   (A0),VECOUT
 JSR      TEST
 MOVE.B   #$0D,VECOUT
 JSR      TEST
 MOVE.B   #'R',VECOUT
 JSR      TEST
 MOVE.B   #'E',VECOUT
 MOVEM.L  (A7)+,D0-D1/A0
 RTS
*********
*         SUBROUTINE GRAPH:
*         PLOTS GRAPH OF PIXEL VALUES ON Y AXIS OF CURSOR
GRAPH:
 MOVEM.L  D0-D4/A0-A3,-(A7)
 MOVE.L   #PAGE1,A1
 MOVE.L   #$19300,D0
BLANKP1:
 MOVE.W   #0,(A1)+
 SUB.L    #1,D0
 BNE      BLANKP1
 MOVE.L   #PAGE2,A2
 MOVE.L   #PAGE1,A3
 CLR.L    D4
 MOVE.W   CUSVALH,D4
 SUB.W    #336,D4
 LSL.W    #1,D4
 ADD.L    D4,A2
 MOVE.W   #403,D0
GRAPH1:
 MOVE.L   A3,A1
 CLR.L    D1
 MOVE.W   (A2),D1
 LSR.W    #4,D1
 LSL.W    #1,D1
 ADD.L    D1,A1
 MOVE.W   #4095,(A1)
 ADD.L    #512,A2
 ADD.L    #512,A3
 SUB.W    #1,D0
 BNE      GRAPH1
 MOVE.B   #1,PAGE
 MOVE.W   #BASECOL1,X
 JSR      SHOW
 MOVEM.L  (A7)+,D0-D4/A0-A3
 RTS
*********
*         VECTRIX COLOR COMMAND
C:
 JSR      TEST  *WAIT FOR EMPTY TRANSMIT BUFFER
 MOVE.B   #'C',VECOUT    *'C' = VECTRIX COLOR COMMAND
 JSR      TEST
 MOVE.B   D0,VECOUT
 ROR.W    #8,D0
 JSR      TEST
 MOVE.B   D0,VECOUT
 ROL.W    #8,D0
 RTS
*********
*         VECTRIX LINE COMMAND
*         USES X,Y
L:
 MOVE.L   D0,-(A7)
 JSR      TEST
 MOVE.B   #'L',VECOUT
 MOVE.W   X,D0
 JSR      TEST
```

continued

```
 MOVE.B    D0,VECOUT
 ROR.W     #8,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 MOVE.W    Y,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROR.W     #8,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 MOVE.L    (A7)+,D0
 RTS
*********
LOG:
 MOVEM.L   D0-D2/A0-A2,-(A7)
 MOVE.L    #PAGE1,A1
 MOVE.L    #PAGE2,A2
 MOVE.L    #LOGDATA,A0
 MOVE.L    #$19300,D0
LOGLOOP:
 MOVE.W    (A2)+,D1
 LSL.W     #1,D1
 MOVE.W    0(A0,D1),D2
 MOVE.W    D2,(A1)+
 SUB.L     #1,D0
 BNE       LOGLOOP
 MOVEM.L   (A7)+,D0-D2/A0-A2
 RTS
*********
FILEPICK:
 MOVEM.L   D0/A0,-(A7)
 MOVE.L    #FCB1,A0
 ADD.L     #8,A0
RANGE:
 JSR       INNOTIME
 CLR.L     D0
 MOVE.B    SYSIO,D0
 CMP.B     #'0',D0
 BLT       RANGE
 CMP.B     #'9',D0
 BLE       INRANGE
 CMP.B     #'A',D0
 BLT       RANGE
 CMP.B     #'Z',D0
 BLE       INRANGE
 BRA       RANGE
INRANGE:
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 MOVE.B    D0,(A0)
 ADD.L     #4,A0
 MOVE.B    #24,D0
FCBZERO:
 MOVE.B    #0,(A0)+
 SUB.B     #1,D0
 BNE       FCBZERO
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
*          RESTORE SUBPROGRAM
*          RESTORE THE IMAGE ON DISK TO PAGE1
RESTORE:
 MOVEM.L   D0-D3/A0-A1,-(A7)
 MOVE.W    #$0F,D0         *CP/M BDOS OPEN FILE FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2              *OPEN OLD IMAGE FILE
 MOVE.L    #FCB1,A0
 MOVE.B    #0,32(A0)
 MOVE.W    #$1A,D0         *CP/M BDOS SET DMA FUNCTION
 MOVE.L    #DMA,D1
 TRAP      #2              *DIRECT DATA TO THIS DMA BUFFER
 MOVE.L    #PAGE1,A0
 MOVE.W    #1612,D2
NFCB:
 MOVE.L    #DMA,A1
 MOVE.B    #128,D3
 MOVE.W    #$14,D0         *CP/M BDOS READ SEQUENTIAL FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2              *READ NEXT 128 BYTE BLOCK TO DISK
NBYTE:
 CLR.L     D1
 MOVE.B    (A1)+,D1
 MOVE.B    D1,(A0)+
 SUB.B     #1,D3
 BNE       NBYTE
 SUB.W     #1,D2
```

```
 BNE       NFCB
 MOVE.W    #$10,D0         *CP/M BDOS CLOSE FILE FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2
 MOVE.W    #$25,D0         *CP/M BDOS RESET DRIVE FUNCTION
 MOVE.W    #2,D1
 TRAP      #2
 MOVEM.L   (A7)+,D0-D3/A0-A1
SAVEDONE:
 RTS
**********
*          SUBROUTINE MAKEDARK
MAKEDARK:
 MOVEM.L   D0-D3/A0-A3,-(A7)
 MOVE.L    #PAGE3,A3
 MOVE.L    #PAGE1,A1
 MOVE.L    #$19300,D0
MD1:
 MOVE.W    (A1)+,(A3)+
 SUB.L     #1,D0
 BNE       MD1
 MOVEM.L   (A7)+,D0-D3/A0-A3
 RTS
*********
*          SUBROUTINE P2TO1
*          MOVE PAGE2 TO PAGE1
P2TO1:
 MOVEM.L   D0-D3/A0-A3,-(A7)
 MOVE.L    #PAGE2,A2
 MOVE.L    #PAGE1,A1
 MOVE.L    #$19300,D0
P2TO1LO:
 MOVE.W    (A2)+,(A1)+
 SUB.L     #1,D0
 BNE       P2TO1LO
 MOVEM.L   (A7)+,D0-D3/A0-A3
 RTS
********
*          SUBROUTINE MAKEFLAT
MAKEFLAT:
 MOVEM.L   D0-D3/A0-A3,-(A7)
 MOVE.L    #PAGE4,A3
 MOVE.L    #PAGE1,A1
 MOVE.L    #$19300,D0
MF1:
 MOVE.W    (A1)+,(A3)+
 SUB.L     #1,D0
 BNE       MD1
 MOVEM.L   (A7)+,D0-D3/A0-A3
 RTS
*********
*          SUBROUTINE DRKFIELD
*          SUBTRACTS DARKFIELD FROM STAR IMAGE
DRKFIELD:
 MOVEM.L   D0-D3/A0-A3,-(A7)
 MOVE.L    #PAGE3,A3
 MOVE.L    #PAGE2,A2
 MOVE.L    #PAGE1,A1
 MOVE.L    #$19300,D0
DFIELD0:
 MOVE.W    (A2)+,D2
 MOVE.W    (A3)+,D3
 CMP.W     #4095,D2
 BEQ       DFIELD1
 SUB.W     D3,D2
 BCC       DFIELD1
 MOVE.W    #0,D2
DFIELD1:
 MOVE.W    D2,(A1)+
 SUB.L     #1,D0
 BNE       DFIELD0
 MOVEM.L   (A7)+,D0-D3/A0-A3
 RTS
*********
*          SUBROUTINE FLTFIELD
*          PUTS TEMPLAT AND RESULT IN PAGE 1
FLTFIELD:
 MOVEM.L   D0-D5/A0-A4,-(A7)
 CLR.L     D1
 CLR.L     D2
 MOVE.L    #PAGE4,A0
 ADD.L     #$19300,A0
 MOVE.L    #$C980,D0
 CLR.L     D5
```

continued

```
FF0:
 MOVE.W    (A0)+,D5
 ADD.L     D5,D1
 SUB.L     #1,D0
 BNE       FF0
 DIVU      #1664,D1
 AND.L     #$0000FFFF,D1
 DIVU      #31,D1
 AND.L     #$0000FFFF,D1
 MOVE.L    #PAGE4,A4
 MOVE.L    #PAGE2,A2
 MOVE.L    #PAGE1,A1
 MOVE.L    #$19300,D0
 MOVE.L    #12,D3
FF1:
 CLR.L     D2
 MOVE.W    (A4)+,D2
 ROL.L     D3,D2
 CMP.W     #0,D1
 BEQ       ZERO0
 DIVU      D1,D2
ZERO0:
 CLR.L     D4
 MOVE.W    (A2)+,D4
 CMP.W     #4095,D4
 BEQ       ZERO1
 ROL.L     D3,D4
 CMP.W     #0,D2
 BEQ       ZERO1
 DIVU      D2,D4
ZERO1:
 MOVE.W    D4,(A1)+
 SUB.L     #1,D0
 BNE       FF1
 MOVEM.L   (A7)+,D0-D5/A0-A4
 RTS
*********
*          SAVE SUBPROGRAM
*          SAVE THE IMAGE IN PAGE1 TO DISK
SAVE:
 MOVEM.L   D0-D3/A0-A1,-(A7)
 MOVE.W    #$13,D0          *CP/M BDOS DELETE FILE FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2               *ERASE ANY OLD FILE BY THIS NAME
 MOVE.W    #$16,D0          *CP/M BDOS MAKE FILE FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2               *CREATE A NEW IMAGE FILE
 MOVE.W    #$1A,D0          *CP/M BDOS SET DMA FUNCTION
 MOVE.L    #DMA,D1
 TRAP      #2               *DIRECT DATA TO THIS DMA BUFFER
 MOVE.L    #PAGE1,A0
 MOVE.W    #1612,D2
NEXTFCB:
 MOVE.L    #DMA,A1
 MOVE.B    #128,D3
NEXTBYTE:
 MOVE.B    (A0)+,(A1)+
 SUB.B     #1,D3
 BNE       NEXTBYTE
 MOVE.W    #$15,D0          *CP/M BDOS WRITE SEQUENTIAL FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2               *WRITE NEXT 128 BYTE BLOCK TO DISK
 SUB.W     #1,D2
 BNE       NEXTFCB
 MOVE.W    #$10,D0          *CP/M BDOS CLOSE FILE FUNCTION
 MOVE.L    #FCB1,D1
 TRAP      #2
 MOVE.W    #$25,D0          *CP/M BDOS RESET DRIVE FUNCTION
 MOVE.W    #2,D1
 TRAP      #2
 MOVEM.L   (A7)+,D0-D3/A0-A1
 RTS
*********
*          SUBPROGRAM WHIRL
*          SPINS COLORS IN WCOLOR THROUGH THE VECTRIX LOOKUP TABLE
WHIRL:
 MOVEM.L   D0-D3/A0-A1,-(A7)
WLOOP1:
 MOVE.L    #WCOLORS,A0
 MOVE.L    #WCOLORS,A1
 ADDA.L    #765,A1
 MOVE.B    (A0)+,(A1)+
 MOVE.B    (A0)+,(A1)+
 MOVE.B    (A0)+,(A1)+
 MOVE.L    #WCOLORS,A0
 MOVE.L    #WCOLORS,A1
```

```
 ADDA.L   #3,A1
 MOVE.W   #765,D0
WLOOP3:
 MOVE.B   (A1)+,(A0)+
 SUB.W    #1,D0
 BNE      WLOOP3
 MOVE.B   #255,D0
 JSR      TEST
 MOVE.B   #'Q',VECOUT
 JSR      TEST
 MOVE.B   #1,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #255,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 MOVE.L   #WCOLORS,A0
WLOOP2:
 MOVE.B   (A0)+,RED
 MOVE.B   (A0)+,GREEN
 MOVE.B   (A0)+,BLUE
 JSR      COLOUT
 SUB.B    #1,D0
 BNE      WLOOP2
 BTST     #1,SYSST
 BNE      WDONE
 BRA      WLOOP1
WDONE:
 MOVEM.L  (A7)+,D0-D3/A0-A1
 RTS
*********
*         GROUPS WILL FIND AND CHARACTERIZE GROUPS OF PIXELS
*         LOOKING AT 403 LINES OF 256 PIXELS EACH
GROUPS:
 MOVEM.L  D0-D7/A0-A6,-(A7)
 JSR      GPINIT
 MOVE.L   #PAGE2,A0
 MOVE.L   #8192,D3
BOTOUT:
 MOVE.W   #0,(A0)+
 SUB.L    #1,D3
 BNE      BOTOUT
 CLR.L    D0
 CLR.L    D1
 CLR.L    D2
 MOVE.L   #PAGE2,A0
 ADD.L    #16384,A0
 MOVE.L   #$19300,D3
 SUB.L    #8192,D3
AVGBASE:
 MOVE.W   (A0)+,D2
 ADD.L    D2,D0
 SUB.L    #1,D3
 BNE      AVGBASE
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.L   #103168,D0
 JSR      STUFFD0
 MOVE.B   #$1C,MATHCMD
 MOVE.B   #$13,MATHCMD
 MOVE.B   #$1E,MATHCMD
 JSR      YANKD0
 MOVE.W   D0,D2
 LSR.W    #1,D0
 ADD.W    D0,D2
 CLR.L    D1
 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A0
 MOVE.L   #$19300,D0
FINDGPS:
 MOVE.W   (A0)+,D1
 CMP.W    D2,D1
 BGT      KEEP            *THIS IS THE CUTOFF DETERMINATION
 MOVE.W   #0,D1
 BRA      OUT2
KEEP:
 MOVE.W   #1,D1
OUT2:
 MOVE.W   D1,(A1)+        *PUTS BINARY IMAGE OF PAGE2 INTO PAGE1
 SUB.L    #1,D0
 BNE      FINDGPS
 MOVE.L   #$19301,D0
 MOVE.L   #PAGE1,A0
```

continued

```
LOOKNXT:
 SUB.L    #1,D0
 BEQ      GPDONE
 MOVE.W   (A0)+,D1
 CMP.W    #1,D1
 BNE      LOOKNXT
 MOVE.L   A0,A1
 SUBA.L   #4,A1
 CLR.L    D1
 CLR.L    D2
 CLR.L    D3
 CLR.L    D4
 MOVE.W   (A1),D1
 SUBA.L   #512,A1
 MOVE.W   (A1)+,D2
 MOVE.W   (A1)+,D3
 MOVE.W   (A1),D4
 CLR.L    D5
 CMP.W    D1,D2
 BGT      GPO1
 EXG      D1,D2
GPO1:
 CMP.W    D2,D3
 BGT      GPO2
 EXG      D2,D3
GPO2:
 CMP.W    D3,D4
 BGT      GPO3
 EXG      D3,D4
GPO3:
 CMP.W    D1,D2
 BGT      GPO4
 EXG      D1,D2
GPO4:
 CMP.W    D2,D3
 BGT      GPO5
 EXG      D2,D3
GPO5:
 CMP.W    D1,D2
 BGT      GPO6
 EXG      D1,D2
GPO6:
 CMP.W    #0,D4
 BEQ      NEWGROUP
 CMP.W    #0,D1
 BEQ      GPOA
 CMP.W    D1,D4
 BEQ      OLDGROUP
 BRA      CONFLICT
GPOA:
 CMP.W    #0,D2
 BEQ      GPOB
 CMP.W    D2,D4
 BEQ      OLDGROUP
 BRA      CONFLICT
GPOB:
 CMP.W    #0,D3
 BEQ      GPOC
 CMP.W    D3,D4
GPOC:
 BEQ      OLDGROUP
 BRA      CONFLICT
*      ASSIGN A NEW GROUP NUMBER TO THIS PIXEL
NEWGROUP:
 MOVE.L   #AREA,A1
 MOVE.L   A1,D1
 ADD.L    #4,A1
 ADD.L    #$4000,D1       *D1 NOW POINTS TO THE END OF 'AREA'
NOTEMPTY:
 CMP.L    A1,D1
 BEQ      FULL
 MOVE.L   (A1)+,D2
 BNE      NOTEMPTY
 SUB.L    #4,A1           *A1 POINTS TO AN EMPTY SPOT IN 'AREA'
 MOVE.L   #1,(A1)
 MOVE.L   A1,D1
 SUB.L    #AREA,D1
 LSR.L    #2,D1           *D1 NOW HOLDS GROUP NUMBER
 MOVE.L   A0,A1
 SUB.L    #2,A1
 MOVE.W   D1,(A1)         *PUT GROUP NUMBER BACK IN HIGH BINARY IMAGE
 MOVE.L   A1,D2
 SUB.L    #PAGE1,D2       *D2 NOW HOLDS OFFSET TO PAGE2
 MOVE.L   #PAGE2,A1
 ADD.L    A1,D2
 MOVE.L   D2,A1
```

```
 CLR.L    D3
 MOVE.W   (A1),D3
 MOVE.L   #BRIGHT,A1
 LSL.L    #2,D1          *MULTIPLY BY 4 FOR LONGWORD TRANSFER
 MOVE.L   D3,0(A1,D1)    *FILLS GROUPth LOCATION IN BRIGHTNESS TABLE
 LSR.L    #1,D1          *DIVIDE BY 2 FOR WORD TRANSFER
 MOVE.L   A0,D3
 SUB.L    #2,D3
 SUB.L    #PAGE1,D3
 DIVU     #512,D3
 MOVE.L   #VMAX,A1
 MOVE.W   D3,0(A1,D1)    *TRANSFER VMAX
 MOVE.L   #VMIN,A1
 MOVE.W   D3,0(A1,D1)    *TRANSFER VMIN
 MOVE.L   #HMAX,A1
 SWAP     D3             *REMAINDER AFTER DIV. =  HORIZ. POSITION
 LSR.W    #1,D3
 MOVE.W   D3,0(A1,D1)    *TRANSFER HMAX
 MOVE.L   #HMIN,A1
 MOVE.W   D3,0(A1,D1)    *TRANSFER HMIN
 BRA      LOOKNXT


OLDGROUP:
 MOVE.L   A0,D1
 SUB.L    #2,D1
 MOVE.L   D1,A1
 MOVE.W   D4,(A1)         *PUT GROUP# IN THIS LOCATION OF PAGE1
 SUB.L    #PAGE1,D1
 ADD.L    #PAGE2,D1
 MOVE.L   D1,A1
 CLR.L    D1
 MOVE.W   (A1),D1
 LSL.L    #2,D4 *MULTIPLY BY FOUR FOR LONGWORD TRANSFER
 MOVE.L   #BRIGHT,A1
 ADD.L    D1,0(A1,D4)    *UPDATE INTEGRATED BRIGHTNESS FOR GROUP
 MOVE.L   #AREA,A1
 MOVE.L   0(A1,D4),D1
 ADD.L    #1,D1
 MOVE.L   D1,0(A1,D4)
 LSR.L    #1,D4          *DIVIDE BY 2 FOR WORD TRANSFER
 MOVE.L   A0,D1
 SUB.L    #2,D1
 SUB.L    #PAGE1,D1
 DIVU     #512,D1
 MOVE.L   #VMAX,A1        *VMIN WAS TAKEN CARE OF WHEN GROUP EST.
 MOVE.W   D1,0(A1,D4)     *WE GO UP, THIS V >= OLD V, SO JUST FILL IN
 SWAP     D1              *THIS LEAVES HORIZONTAL POS. IN LOW WORD
 LSR.W    #1,D1
 MOVE.L   #HMAX,A1
 MOVE.W   0(A1,D4),D2
 CMP.W    D1,D2
 BGT      OLD1
 MOVE.W   D1,0(A1,D4)
OLD1:
 MOVE.L   #HMIN,A1
 MOVE.W   0(A1,D4),D2
 CMP.W    D1,D2
 BLT      OLD2
 MOVE.W   D1,0(A1,D4)
OLD2:
 BRA      LOOKNXT

CONFLICT:
 CMP.W    #0,D1
 BNE      LINKUP
 EXG      D1,D2
 CMP.W    #0,D1
 BNE      LINKUP
 EXG      D1,D3
LINKUP:
 LSL.W    #2,D1           *MULTIPLY BY FOUR FOR LONGWORD TRANSFER
 LSL.W    #2,D4
 MOVE.L   #AREA,A1
 MOVE.L   0(A1,D4),D2
 MOVE.L   #0,0(A1,D4)
 ADD.L    D2,0(A1,D1)
 MOVE.L   #BRIGHT,A1
 MOVE.L   0(A1,D4),D2
 MOVE.L   #0,0(A1,D4)
 ADD.L    D2,0(A1,D1)
 LSR.W    #1,D1
 LSR.W    #1,D4           *DIVIDE BY TWO FOR WORD TRANSFERS
 MOVE.L   #VMAX,A1
 MOVE.L   0(A1,D1),D2
 MOVE.L   0(A1,D4),D3
```

continued

```
 MOVE.L   #0,0(A1,D4)
 CMP.L    D2,D3
 BGT      LINK1
 EXG      D2,D3
LINK1:
 MOVE.L   D3,0(A1,D1)
 MOVE.L   #HMAX,A1
 MOVE.L   0(A1,D1),D2
 MOVE.L   0(A1,D4),D3
 MOVE.L   #0,0(A1,D4)
 CMP.L    D2,D3
 BGT      LINK2
 EXG      D2,D3
LINK2:
 MOVE.L   D3,0(A1,D1)
 MOVE.L   #VMIN,A1
 MOVE.L   0(A1,D1),D2
 MOVE.L   0(A1,D4),D3
 MOVE.L   #$1FF,0(A1,D4)
 CMP.L    D2,D3
 BLT      LINK3
 EXG      D2,D3
LINK3:
 MOVE.L   D3,0(A1,D1)
 MOVE.L   #HMIN,A1
 MOVE.L   0(A1,D1),D2
 MOVE.L   0(A1,D4),D3
 MOVE.L   #$FF,0(A1,D4)
 CMP.L    D2,D3
 BLT      LINK4
 EXG      D2,D3
LINK4:
 MOVE.L   D3,0(A1,D1)
 EXG      D1,D4           *PUT GROUP NUMBER WHERE OLDGROUP CAN FIND IT
 LSR.L    #1,D4           *AND MAKE IT THE RIGHT SIZE
 BRA      OLDGROUP

FULL:
 MOVE.L   #ERRMSG,A1
 JSR      MSGOUT
 MOVEM.L  (A7)+,D0-D7/A0-A6
 RTS

GPDONE:
 MOVE.L   #$19300,D0
 MOVE.L   #PAGE1,A0
MULTPAG1:
 MOVE.W   (A0),D1
 LSL.W    #4,D1
 MOVE.W   D1,(A0)+
 SUB.L    #1,D0
 BNE      MULTPAG1
 MOVEM.L  (A7)+,D0-D7/A0-A6
 RTS
*********
*        GPINIT SUBROUTINE
*        THIS INITIALIZES H&V MAX&MIN, BRIGHT, AND AREA FOR GROUPS PROGRAM
GPINIT:
 MOVE.L   #256,D0
 MOVE.L   #HMAX,A0
GPLOOP2:
 CLR.W    (A0)+           *SET INITIAL HMAXs TO 0
 DBF      D0,GPLOOP2
*      VERTICAL MAX INIT
 MOVE.L   #256,D0
 MOVE.L   #VMAX,A0
GPLOOP3:
 CLR.W    (A0)+           *SET INITIAL VMAXs TO 0
 DBF      D0,GPLOOP3
*      HORIZONTAL MIN INIT
 MOVE.L   #256,D0
 MOVE.L   #HMIN,A0
GPLOOP4:
 MOVE.W   #$100,(A0)+     *LARGER THAN ANY REAL HMIN
 DBF      D0,GPLOOP4
*      VERTICAL MIN INIT
 MOVE.L   #256,D0
 MOVE.L   #VMIN,A0
GPLOOP5:
 MOVE.W   #$1FF,(A0)+     *LARGER THAN ANY REAL VMIN
 DBF      D0,GPLOOP5
*      BRIGHTNESS INIT
 MOVE.L   #256,D0
 MOVE.L   #BRIGHT,A0
GPLOOP6:
 CLR.L    (A0)+
```

```
 DBF      D0,GPLOOP6
*      AREA INIT
 MOVE.L   #256,D0
 MOVE.L   #AREA,A0
GPLOOP7:
 CLR.L    (A0)+
 DBF      D0,GPLOOP7
 RTS   *TO GROUPS
*********
*         HISTCURS
*         HISTOGRAM CURSOR
HISTCURS:
 MOVEM.L  D0-D6/A0-A1,-(A7)
 MOVE.W   CUSVALR,D1
 MOVE.W   CUSVALL,D2
KEYINH:
 JSR      HCWHITE
 JSR      HCBLACK
 BTST     #1,SYSST        *KEY PRESSED?
 BEQ      KEYINH
 MOVE.B   SYSIO,D0
 CMP.B    #$08,D0         *LEFT ARROW ?
 BEQ      RLEFT
 CMP.B    #$0C,D0         *RIGHT ARROW ?
 BEQ      RRIGHT
 CMP.B    #$0B,D0         *UP ARROW ?
 BEQ      LRIGHT
 CMP.B    #$16,D0         *DOWN ARROW ?
 BEQ      LLEFT
 CMP.B    #$0D,D0         *CARRIGE RETURN ?
 BEQ      STRETCH
 BRA      KEYINH
RRIGHT:
 CMP.W    #590,D1
 BGT      KEYINH
 ADD.W    #1,D1
 MOVE.W   D1,CUSVALR
 BRA      KEYINH
RLEFT:
 CMP.W    D2,D1
 BEQ      LLEFT
 SUB.W    #1,D1
 MOVE.W   D1,CUSVALR
 BRA      KEYINH
LRIGHT:
 CMP.W    D2,D1
 BEQ      RRIGHT
 ADD.W    #1,D2
 MOVE.W   D2,CUSVALL
 BRA      KEYINH
LLEFT:
 CMP.W    #337,D2
 BLT      KEYINH
 SUB.W    #1,D2
 MOVE.W   D2,CUSVALL
 BRA      KEYINH
HCWHITE:
 JSR      TEST
 MOVE.B   #'C',VECOUT
 JSR      TEST
 MOVE.B   #$FF,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 BRA      CURSOUT
HCBLACK:
  JSR      TEST
 MOVE.B   #'C',VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
CURSOUT:
 JSR      TEST
 MOVE.B   #'M',VECOUT
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROR.W    #8,D1
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROL.W    #8,D1
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
```

continued

```
 JSR       TEST
 MOVE.B    #'L',VECOUT
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROR.W     #8,D1
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROL.W     #8,D1
 JSR       TEST
 MOVE.B    #3,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #'M',VECOUT
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROR.W     #8,D2
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROL.W     #8,D2
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #'L',VECOUT
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROR.W     #8,D2
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROL.W     #8,D2
 JSR       TEST
 MOVE.B    #3,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 RTS
STRETCH:
 MOVE.L    #$19300,D0
 MOVE.L    #PAGE2,A0
 MOVE.L    #PAGE1,A1
 SUB.W     #336,D1
 SUB.W     #336,D2
 LSL.W     #4,D1
 LSL.W     #4,D2
STALOOP:
 CLR.L     D3
 MOVE.W    (A0)+,D3
 CMP.W     D2,D3
 BGT       STA1
 MOVE.W    #0,D3
 BRA       STAOUT
STA1:
 CMP.W     D1,D3
 BLT       STA2
 MOVE.W    #4095,D3
 BRA       STAOUT
STA2:
 MOVE.W    D2,D4
 MOVE.W    D1,D5
 SUB.W     D4,D3
 SUB.W     D4,D5
 MULU      #4095,D3
 DIVU      D5,D3
STAOUT:
   MOVE.W    D3,(A1)+
 SUB.L     #1,D0
 BNE       STALOOP
CURSDONE:
 MOVEM.L   (A7)+,D0-D6/A0-A1
 RTS
*********
*          SCRNVAL
SCRNVAL:
 MOVEM.L   D0-D7/A0,-(A7)
 MOVE.L    #WCOLORS,A0
 MOVE.W    CUSVALR,D1
 MOVE.W    CUSVALL,D2
 JSR       LETSEE
KEYINS:
 JSR       VALWHITE
 JSR       VALBLACK
 BTST      #1,SYSST
 BEQ       KEYINS
 JSR       NOSEE
 MOVE.B    SYSIO,D0
```

```
 CMP.B     #$08,D0
 BEQ       LLEFTS
 CMP.B     #$0C,D0
 BEQ       LRIGHTS
 CMP.B     #$0D,D0
 BEQ       VALDONE
 BRA       KEYINS

RRIGHTS:
 CMP.W     #590,D1
 BGT       KEYINS
 ADD.W     #1,D1
 MOVE.W    D1,CUSVALR
 JSR       LETSEE
 BRA       KEYINS
LRIGHTS:
 CMP.W     D1,D2
 BEQ       RRIGHTS
 ADD.W     #1,D2
 MOVE.W    D2,CUSVALL
 JSR       LETSEE
 BRA       KEYINS
LLEFTS:
 CMP.W     #337,D2
 BLT       KEYINS
 SUB.W     #1,D2
 MOVE.W    D2,CUSVALL
 JSR       LETSEE
 BRA       KEYINS

VALWHITE:
 JSR       TEST
 MOVE.B    #'C',VECOUT
 JSR       TEST
 MOVE.B    #$FF,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 BRA       CURSVAL
VALBLACK:
 JSR       TEST
 MOVE.B    #'C',VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
CURSVAL:
 JSR       TEST
 MOVE.B    #'M',VECOUT
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROR.W     #8,D2
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROL.W     #8,D2
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #'L',VECOUT
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROR.W     #8,D2
 JSR       TEST
 MOVE.B    D2,VECOUT
 ROL.W     #8,D2
 JSR       TEST
 MOVE.B    #3,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 RTS
LETSEE:
 MOVE.W    D2,D3
 CLR.L     D7
 SUB.W     #336,D3
 MOVE.W    D3,D7
 MULU      #3,D3
 AND.L     #$0000FFFF,D3
 MOVE.B    #0(A0,D3),D4
 MOVE.B    #1(A0,D3),D5
 MOVE.B    #2(A0,D3),D6
 JSR       TEST
 MOVE.B    #'Q',VECOUT
 JSR       TEST
```

continued

```
  MOVE.B    D7,VECOUT
  JSR       TEST
  MOVE.B    #0,VECOUT
  JSR       TEST
  MOVE.B    #1,VECOUT
  JSR       TEST
  MOVE.B    #0,VECOUT
  MOVE.B    #255,RED
  MOVE.B    #0,GREEN
  MOVE.B    #255,BLUE
  JSR       COLOUT
  JSR       TEST
  MOVE.B    #'C',VECOUT
  JSR       TEST
  MOVE.B    #$FF,VECOUT
  JSR       TEST
  MOVE.B    #0,VECOUT
  JSR       TEST
  MOVE.B    #'R',VECOUT
  JSR       TEST
  MOVE.B    #'A',VECOUT
 OUTVAL:
  MOVE.W    #600,X
  MOVE.W    #40,Y
  JSR       TEST
  JSR       M
  JSR       TEST
  MOVE.B    #'$',VECOUT
 OUTVAL1:
  JSR       TEST
  MOVE.B    ' ',VECOUT
  MOVE.W    D2,D3
  SUB.W     #336,D3
  AND.W     #$00F0,D3
  ROR.W     #4,D3
  ADD.W     #$30,D3
  CMP.W     #$3A,D3
  BLT       OUTVAL2
  ADD.W     #7,D3
 OUTVAL2:
  JSR       TEST
  MOVE.B    D3,VECOUT
  MOVE.W    D2,D3
  SUB.W     #336,D3
  AND.W     #$000F,D3
  ADD.W     #$30,D3
  CMP.W     #$3A,D3
  BLT       OUTVAL3
  ADD.W     #7,D3
 OUTVAL3:
  JSR       TEST
  MOVE.B    D3,VECOUT
  JSR       TEST
  MOVE.B    #$0D,VECOUT
  JSR       TEST
  MOVE.B    #'R',VECOUT
  JSR       TEST
  MOVE.B    #'E',VECOUT
  RTS
NOSEE:
  CLR.L     D7
  MOVE.W    D2,D7
  SUB.W     #336,D7
  JSR       TEST
  MOVE.B    #'Q',VECOUT
  JSR       TEST
  MOVE.B    D7,VECOUT
  JSR       TEST
  MOVE.B    #0,VECOUT
  JSR       TEST
  MOVE.B    #1,VECOUT
  JSR       TEST
  MOVE.B    #0,VECOUT
  MOVE.B    D4,RED
  MOVE.B    D5,GREEN
  MOVE.B    D6,BLUE
  JSR       COLOUT
  RTS
  VALDONE:
  MOVEM.L   (A7)+,D0-D7/A0
  RTS
*********
THREED:
  MOVEM.L   D0-D4/A0-A3,-(A7)
  MOVE.L    #PAGE1,A1
  MOVE.L    #$19300,D0
```

```
TD0:
 MOVE.W    #$1000,(A1)+
 SUB.L     #1,D0
 BNE       TD0
 MOVE.L    #PAGE1,A0
 MOVE.L    #PAGE1,A1
 ADD.L     #510,A1
 MOVE.L    #PAGE2,A2
 ADD.L     #510,A2
 CLR.L     D1
 MOVE.L    #403,D2
 MOVE.L    #256,D3
TRELOP1:
 MOVE.L    #0,D0
 MOVE.W    (A2),D1
 AND.W     #$0FFF,D1
 LSR.W     #6,D1
 ADD.W     #1,D1
 MOVE.L    A1,A3
TRELOP2:
 MOVE.W    (A3),D4
 CMP.W     #$1000,D4
 BNE       NEXTTP
 MOVE.W    D0,(A3)
NEXTTP:
 SUB.L     #2,A3
 CMP.L     A3,A0
 BEQ       TREROND
 ADD.L     #64,D0
 SUB.W     #1,D1
 BNE       TRELOP2

TREROND:
 ADD.L     #512,A0
 ADD.L     #512,A1
 ADD.L     #512,A2
 SUB.L     #1,D2
 BNE       TRELOP1
 MOVE.L    #403,D2
 SUB.L     #206336,A0
 SUB.L     #206338,A1
 SUB.L     #206338,A2
 SUB.L     #1,D3
 BNE       TRELOP1
 MOVE.L    #PAGE1,A1
 MOVE.L    #$19300,D0
TD1:
 MOVE.W    (A1),D1
 CMP.W     #$1000,D1
 BNE       TD2
 MOVE.W    #0,D1
TD2:
 MOVE.W    D1,(A1)+
 SUB.L     #1,D0
 BNE       TD1
 MOVEM.L   (A7)+,D0-D4/A0-A3
 RTS
*********
*          MAGNIFY ROUTINE
*          MAGNIFY BY 2 THE IMAGE IN PAGE 2 AND PUTS IT IN PAGE ONE
MAGNIFY:
 MOVEM.L   D0-D4/A0-A5,-(A7)
 JSR       TEST
 MOVE.B    #'B',VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #1,VECOUT
 JSR       TESTOUT
KEYINM:
 JSR       WHITEONE
 JSR       BLACKONE
 BTST      #1,SYSST
 BEQ       KEYINM
 MOVE.B    SYSIO,D0
 CMP.B     #$08,D0
 BEQ       LEFT
 CMP.B     #$0C,D0
 BEQ       RIGHT
 CMP.B     #$0B,D0
 BEQ       UP
 CMP.B     #$16,D0
 BEQ       DOWN
 CMP.B     #$0D,D0
```

continued

```
 BEQ       DOMAG
 BRA       KEYINM
RIGHT:
 MOVE.W    FRAMEX,D0
 CMP.W     #464,D0
 BGT       KEYINM
 ADD.W     #1,D0
 MOVE.W    D0,FRAMEX
 BRA       KEYINM
LEFT:
 MOVE.W    FRAMEX,D0
 CMP.W     #336,D0
 BLT       KEYINM
 SUB.W     #1,D0
 MOVE.W    D0,FRAMEX
 BRA       KEYINM
UP:
 MOVE.W    FRAMEY,D0
 CMP.W     #271,D0
 BGT       KEYINM
 ADD.W     #1,D0
 MOVE.W    D0,FRAMEY
 BRA       KEYINM
DOWN:
 MOVE.W    FRAMEY,D0
 CMP.W     #70,D0
 BLT       KEYINM
 SUB.W     #1,D0
 MOVE.W    D0,FRAMEY
 BRA        KEYINM
WHITEONE:
 JSR       TEST
 MOVE.B    #'C',VECOUT
 JSR       TEST
 MOVE.B    #$FF,VECOUT
 JSR       TEST
 MOVE.B    #1,VECOUT
 BRA       FRAME
BLACKONE:
 JSR       TEST
 MOVE.B    #'C',VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
 JSR       TEST
 MOVE.B    #0,VECOUT
FRAME:
 MOVE.W    FRAMEX,D0
 MOVE.W    FRAMEY,D1
 JSR       TEST
 MOVE.B    #'P',VECOUT
 JSR       TEST
 MOVE.B    #$4,VECOUT
 JSR       TEST
 MOVE.B    #$0,VECOUT
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROR.W     #8,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROL.W     #8,D0
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROR.W     #8,D1
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROL.W     #8,D1
 ADD.W     #128,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROR.W     #8,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROL.W     #8,D0
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROR.W     #8,D1
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROL.W     #8,D1
 ADD.W     #200,D1
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROR.W     #8,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROL.W     #8,D0
```

```
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROR.W     #8,D1
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROL.W     #8,D1
 SUB.W     #128,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROR.W     #8,D0
 JSR       TEST
 MOVE.B    D0,VECOUT
 ROL.W     #8,D0
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROR.W     #8,D1
 JSR       TEST
 MOVE.B    D1,VECOUT
 ROL.W     #8,D1
 RTS
DOMAG:
 CLR.L     D0
 CLR.L     D1
 MOVE.W    FRAMEY,D1
 SUB.W     #69,D1
 MULU      #512,D1
 MOVE.W    FRAMEX,D0
 SUB.W     #335,D0
 ROL.W     #1,D0
 ADD.W     D0,D1
 ADD.L     #PAGE2,D1
 MOVE.L    D1,A0
 MOVE.W    #200,D3
 MOVE.L    #PAGE1,D0
NUTLINE:
  MOVE.L   D0,A1
 ADD.L     #512,D0
 MOVE.L    D0,A2
 ADD.L     #512,D0
 MOVE.W    #128,D4
THISLINE:
 MOVE.W    (A0)+,D2
 MOVE.W    D2,(A1)+
 MOVE.W    D2,(A2)+
 MOVE.W    D2,(A1)+
 MOVE.W    D2,(A2)+
 SUB.W     #1,D4
 BNE       THISLINE
 ADD.L     #256,A0
 SUB.W     #1,D3
 BNE       NUTLINE
MAGDONE:
 JSR       TEST
 MOVE.B    #'B',VECOUT
 JSR       TEST
 MOVE.B    #$FF,VECOUT
 JSR       TEST
 MOVE.B    #$1,VECOUT
 MOVEM.L   (A7)+,D0-D4/A0-A5
 RTS
*********
COLOR:
 MOVEM.L   D0-D2,-(A7)
COLOOP:
 JSR       TESTIN
 CLR.L     D0
 MOVE.B    SYSIO,D0
MXC1:
 CMP.B     #'1',D0
 BNE       MXC2
 JSR       GREY
MXC2:
 CMP.B     #'2',D0
 BNE       MXC3
 JSR       SPECTRUM
MXC3:
 CMP.B     #'3',D0
 BNE       MXC4
 JSR       CONTOUR
MXC4:
 CMP.B     #'4',D0
 BNE       MXC5
 JSR       WINIT
```

continued

```
MXC5:
 CMP.B     #$0D,D0
 BNE       COLOOP
COLOROUT:
 MOVEM.L   (A7)+,D0-D2
 RTS
*********
*          CONTOUR SUBROUTINE
*          GIVES GOOD COLOR VARIATION FOR SMALL CHANGE IN PIXEL VALUE
CONTOUR:
 MOVEM.L    D0/A0,-(A7)
 MOVE.L    #WCOLORS,A0
 CLR.L     D0
 MOVE.W    #16,D0
NXTCHUNK:
 MOVE.B    #$AA,RED
 MOVE.B    #0,GREEN
 MOVE.B    #$55,BLUE
 JSR       COLFILL
 MOVE.B    #$AA,RED
 MOVE.B    #0,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #$FF,RED
 MOVE.B    #0,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #$FF,RED
 MOVE.B    #$70,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #$AA,RED
 MOVE.B    #$70,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #$FF,RED
 MOVE.B    #$AA,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #$FF,RED
 MOVE.B    #$FF,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #$AA,RED
 MOVE.B    #$FF,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #0,RED
 MOVE.B    #$AA,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #0,RED
 MOVE.B    #$FF,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #0,RED
 MOVE.B    #$FF,GREEN
 MOVE.B    #$AA,BLUE
 JSR       COLFILL
 MOVE.B    #0,RED
 MOVE.B    #$FF,GREEN
 MOVE.B    #$FF,BLUE
 JSR       COLFILL
 MOVE.B    #0,RED
 MOVE.B    #$AA,GREEN
 MOVE.B    #$FF,BLUE
 JSR       COLFILL
 MOVE.B    #0,RED
 MOVE.B    #$55,GREEN
 MOVE.B    #$FF,BLUE
 JSR       COLFILL
 MOVE.B    #$AA,RED
 MOVE.B    #0,GREEN
 MOVE.B    #$FF,BLUE
 JSR       COLFILL
 MOVE.B    #$FF,RED
 MOVE.B    #0,GREEN
 MOVE.B    #$FF,BLUE
 JSR       COLFILL
 SUB.B     #1,D0
 BNE       NXTCHUNK
 MOVE.L    #WCOLORS,A0
 MOVE.B    #0,RED
 MOVE.B    #0,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
```

```
 JSR       COLSEND
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
*          WRITES COLOR TO VECTRIX LUT
WINIT:
 MOVEM.L   D0-D5/A0-A1,-(A7)
 MOVE.L    #WCOLORS,A0
 MOVE.B    #0,RED
 MOVE.B    #0,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.B    #0,D0
CGRNUP:
 ADD.B     #1,D0
 CMP.B     #64,D0
 BEQ       CTOPGRN
 JSR       COLFILL
 ADD.B     #4,GREEN
 BRA       CGRNUP
CTOPGRN:
 MOVE.B    #$00,GREEN
 JSR       COLFILL
CBLUEUP:
 ADD.B     #1,D0
 CMP.B     #128,D0
 BEQ       CTOPBLUE
 JSR       COLFILL
 ADD.B     #4,BLUE
 BRA       CBLUEUP
CTOPBLUE:
 MOVE.B    #$00,BLUE
 JSR       COLFILL
CREDUP:
 ADD.B     #1,D0
 CMP.B     #192,D0
 BEQ       CTOPRED
 JSR       COLFILL
 ADD.B     #4,RED
 BRA       CREDUP
CTOPRED:
 MOVE.B    #$00,RED
 JSR       COLFILL
CGREYUP:
 ADD.B     #1,D0
 CMP.B     #255,D0
 BEQ       CTOPGREY
 JSR       COLFILL
 ADD.B     #4,RED
 ADD.B     #4,GREEN
 ADD.B     #4,BLUE
 BRA       CGREYUP
CTOPGREY:
 MOVE.B    #$FF,RED
 MOVE.B    #$FF,GREEN
 MOVE.B    #$FF,BLUE
 JSR       COLFILL
 JSR       COLSEND
 MOVEM.L   (A7)+,D0-D5/A0-A1
 RTS
*********
*          WRITES SPECTRUM SCALE TO THE WCOLOR STORAGE AREA
SPECTRUM:
 MOVEM.L   D0/A0,-(A7)
 MOVE.L    #WCOLORS,A0
 MOVE.B    #0,RED
 MOVE.B    #0,GREEN
 MOVE.B    #0,BLUE
 JSR       COLFILL
 MOVE.L    #1,D0
 MOVE.B    #127,RED
RUP1:
 ADD.B     #4,RED
 JSR       COLFILL
 ADD.L     #1,D0
 CMP.L     #33,D0
 BNE       RUP1
 MOVE.B    #251,RED
WGRNUP1:
 ADD.B     #4,GREEN
 SUB.B     #4,RED
 JSR       COLFILL
 ADD.L     #1,D0
```

continued

```
  CMP.L    #64,D0
  BNE      WGRNUP1
YELLOW1:
  ADD.B    #4,GREEN
  ADD.B    #4,RED
  JSR      COLFILL
  ADD.L    #1,D0
  CMP.L    #96,D0
  BNE      YELLOW1
  MOVE.B   #255,RED
  MOVE.B   #255,GREEN
YELLOW2:
  SUB.B    #8,RED
  JSR      COLFILL
  ADD.L    #1,D0
  CMP.L    #127,D0
  BNE      YELLOW2
WBLUEUP:
  ADD.B    #8,BLUE
  JSR      COLFILL
  ADD.L    #1,D0
  CMP.L    #159,D0
  BNE      WBLUEUP
  MOVE.B   #255,BLUE
BLUEUP2:
  SUB.B    #5,GREEN
  JSR      COLFILL
  ADD.L    #1,D0
  CMP.L    #207,D0
  BNE      BLUEUP2
  MOVE.B   #0,GREEN
RUP2:
  ADD.B    #5,RED
  SUB.B    #3,BLUE
  JSR      COLFILL
  ADD.L    #1,D0
  CMP.L    #255,D0
  BNE      RUP2
WWHITE:
  MOVE.B   #$FF,RED
  MOVE.B   #$FF,GREEN
  MOVE.B   #$FF,BLUE
  JSR      COLFILL
  JSR      COLSEND
  MOVEM.L  (A7)+,D0/A0
  RTS
*********
*          GREY SUBROUTINE
*          WRITES GREY SCALE TO THE BOTTOM 256 LOCATIONS IN THE VECTRIX
*          COLOR LOOKUP TABLE
GREY:
  MOVEM.L  D0/A0,-(A7)
  MOVE.L   #WCOLORS,A0
  CLR.L    D0
GREYOUT:
  MOVE.B   D0,RED
  MOVE.B   D0,GREEN
  MOVE.B   D0,BLUE
  JSR      COLFILL
  ADD.B    #1,D0
  BNE      GREYOUT
  JSR      COLSEND
  MOVEM.L  (A7)+,D0/A0
  RTS
*********
COLSEND:
  MOVEM.L  D0/A0,-(A7)
  JSR      TEST
  MOVE.B   #'Q',VECOUT
  JSR      TEST
  MOVE.B   #0,VECOUT
  JSR      TEST
  MOVE.B   #0,VECOUT
  JSR      TEST
  MOVE.B   #0,VECOUT
  JSR      TEST
  MOVE.B   #1,VECOUT
  MOVE.L   #256,D0
  MOVE.L   #WCOLORS,A0
COLSEND0:
  MOVE.B   (A0)+,RED
  MOVE.B   (A0)+,GREEN
  MOVE.B   (A0)+,BLUE
  JSR      COLOUT
  SUB.L    #1,D0
  BNE      COLSEND0
```

```
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
COLOUT:
 BSR       TEST
 MOVE.B    RED,VECOUT
 BSR       TEST
 MOVE.B    GREEN,VECOUT
 BSR       TEST
 MOVE.B    BLUE,VECOUT
 RTS
*********
COLFILL:
 MOVE.B    RED,(A0)+
 MOVE.B    GREEN,(A0)+
 MOVE.B    BLUE,(A0)+
 RTS
*********
*          REPLACES BAD PIXELS WITH MEDIAN
PIXELFIX:
 MOVEM.L   D0-D1/A0,-(A7)
 JSR       P2TO1
 MOVE.L    #234,D0
 ASL.L     #5,D0
 ASL.L     #4,D0
 MOVE.W    #256,D1
FIXLINE1:
 JSR       MEDIAND0
 ADD.L     #2,D0
 SUB.W     #1,D1
 BNE       FIXLINE1
 MOVE.L    #BADMAP,A0
READMAP:
 MOVE.L    (A0)+,D0
 CMP.L     #$FFFFFFFF,D0
 BEQ       LASTPIX
 JSR       MEDIAND0
 BRA       READMAP
LASTPIX:
 MOVEM.L   (A7)+,D0-D1/A0
 RTS
*********
*          SUBROUTINE MAPMAKER
*          USES PAGE 2 BIT MAP TO LOCATE AND STORE BAD PIXELS
MAPMAKER:
 MOVEM.L   D0-D7/A0-A6,-(A7)
 MOVE.L    #PAGE2,A2
 MOVE.L    #BADMAP,A0
 MOVE.L    #4094,D1
 MOVE.L    #$19300,D0
BADLOOP:
 MOVE.L    A2,A1
 MOVE.W    (A2)+,D2
 CMP.W     #4095,D2
 BNE       BADOVER
 SUB.L     #PAGE2,A1
 MOVE.L    A1,(A0)+
 SUB.L     #1,D1
 BEQ       BADDONE
BADOVER:
 SUB.L     #1,D0
 BNE       BADLOOP
BADDONE:
 MOVE.L    #$FFFFFFFF,(A0)
 MOVEM.L   (A7)+,D0-D7/A0-A6
 RTS
*********
*          REPLACES PIXEL AT D0 WITH 3 X 3 MEDIAN
MEDIAND0:
 MOVEM.L   D0-D7/A0-A6,-(A7)
 MOVE.L    #PAGE1,A6
 MOVE.L    #PAGE2,D2
 ADD.L     D0,A6
 ADD.L     D0,D2
 SUB.L     #2,D2
 SUB.L     #512,D2
 MOVE.L    D2,A3
 ADD.L     #512,D2
 MOVE.L    D2,A4
 ADD.L     #512,D2
 MOVE.L    D2,A5
 MOVE.W    (A3)+,D1
 MOVE.W    (A3)+,D2
```

continued

```
 MOVE.W   (A3)+,D3
 MOVE.W   (A4)+,D4
 MOVE.W   (A4)+,D5
 MOVE.W   (A4)+,D6
 MOVE.W   (A5)+,D7
 MOVE.W   D7,A0
 MOVE.W   (A5)+,D7
 MOVE.W   D7,A1
 MOVE.W   (A5)+,D7
 MOVE.B    #5,D0
E0:
 CMP       D1,D2
 BGT       E1
 EXG       D1,D2
E1:
 CMP       D2,D3
 BGT       E2
 EXG       D2,D3
E2:
 CMP       D3,D4
 BGT       E3
 EXG       D3,D4
E3:
 CMP       D4,D5
 BGT       E4
 EXG       D4,D5
E4:
 CMP       D5,D6
 BGT       E5
 EXG       D5,D6
E5:
 CMP       D6,D7
 BGT       E6
 EXG       D6,D7
E6:
 CMP       D7,A0
 BGT       E7
 EXG       D7,A0
E7:
 CMP       A0,A1
 BGT       E8
 EXG       A0,A1
E8:
 SUB.B     #1,D0
 BNE       E0
 MOVE.W    D5,(A6)
 MOVEM.L   (A7)+,D0-D7/A0-A6
 RTS
*********
*          STAROUT 5 X 5 PIXEL MEDIAN
STAROUT5:
 MOVEM.L   D0-D7/A0-A6,-(A7)
 MOVE.L    #SORTDATA,A5
 MOVE.W    #4096,D0
INITSTAR:
 MOVE.W    #0,(A5)+
 SUB.W     #1,D0
 BNE       INITSTAR
 MOVE.L    #PAGE2,A2
 MOVE.L    #$19300,D0
 SUB.L     #1030,A2
 MOVE.L    #5,D7
INITST1:
 MOVE.L    #5,D6
INITST2:
 MOVE.L    #SORTDATA,A5
 MOVE.L    #0,D1
 MOVE.W    (A2)+,D1
 ASL.L     #1,D1
 ADD.L     D1,A5
 ADD.W     #1,(A5)
 SUB.L     #1,D6
 BNE       INITST2
 ADD.L     #502,A2
 SUB.L     #1,D7
 BNE       INITST1

 MOVE.L    #SORTDATA,A4
 MOVE.L    #0,D3
 MOVE.L    #0,D4
SORTSTA:
 ADD.W     (A4)+,D4
 ADD.W     #1,D3
 CMP.W     #13,D4
 BLT       SORTSTA
 SUB.W     #1,D3
```

```
  SUB.L    #2,A4

  MOVE.L   #PAGE1,A1
  MOVE.L   #PAGE2,A2
  SUB.L    #1030,A2
  MOVE.L   #PAGE2,A3
  SUB.L    #1020,A3
STALOP1:
  MOVE.L   #SORTDATA,A5
  MOVE.W   #5,D7
STALOP2:
  MOVE.L   #0,D1
  MOVE.W   (A2),D1
  CMP.W    D1,D3
  BLT      STARLO1
  SUB.W    #1,D4
STARLO1:
  MOVE.L   A5,A6
  ASL.L    #1,D1
  ADD.L    D1,A6
  SUB.W    #1,(A6)
  ADD.L    #512,A2
  MOVE.L   #0,D1
  MOVE.W   (A3),D1
  CMP.W    D1,D3
  BLT      STARLO2
  ADD.W    #1,D4
STARLO2:
  MOVE.L   A5,A6
  ASL.L    #1,D1
  ADD.L    D1,A6
  ADD.W    #1,(A6)
  ADD.L    #512,A3
  SUB.W    #1,D7
  BNE      STALOP2
  SUB.L    #2558,A2
  SUB.L    #2558,A3
  CMP.W    #13,D4
  BEQ      FOUN
  BGT      LOWE
HIGHE:
  ADD.L    #2,A4
  ADD.W    #1,D3
  ADD.W    (A4),D4
  CMP.W    #13,D4
  BLT      HIGHE
  BRA      FOUN
LOWE:
  SUB.W    (A4),D4
  SUB.L    #2,A4
  SUB.W    #1,D3
  CMP.W    #13,D4
  BGT      LOWE
  ADD.L    #2,A4
  ADD.W    #1,D3
  ADD.W    (A4),D4
FOUN:
  MOVE.L   A4,A6
  SUB.L    A5,A6
  MOVE.L   A6,D1
  LSR.W    #1,D1
  MOVE.W   D1,(A1)+
  SUB.L    #1,D0
  BNE      STALOP1
  MOVEM.L  (A7)+,D0-D7/A0-A6
  RTS
*********
*         STAROUT 15 X 15 PIXEL MEDIAN
STAROUT:
  MOVEM.L  D0-D7/A0-A6,-(A7)
  MOVE.L   #SORTDATA,A5
  MOVE.W   #4096,D0
INITSTR0:
  MOVE.W   #0,(A5)+
  SUB.W    #1,D0
  BNE      INITSTR0
  MOVE.L   #PAGE2,A2
  MOVE.L   #$19300,D0
  SUB.L    #3600,A2
  MOVE.L   #15,D7
INITSTR1:
  MOVE.L   #15,D6
```

continued

```
INITSTR2:
 MOVE.L   #SORTDATA,A5
 MOVE.L   #0,D1
 MOVE.W   (A2)+,D1
 ASL.L    #1,D1
 ADD.L    D1,A5
 ADD.W    #1,(A5)
 SUB.L    #1,D6
 BNE      INITSTR2
 ADD.L    #482,A2
 SUB.L    #1,D7
 BNE      INITSTR1

 MOVE.L   #SORTDATA,A4
 MOVE.L   #0,D3
 MOVE.L   #0,D4
SORTSTAR:
 ADD.W    (A4)+,D4
 ADD.W    #1,D3
 CMP.W    #111,D4
 BLT      SORTSTAR
 SUB.W    #1,D3
 SUB.L    #2,A4

 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A2
 SUB.L    #3600,A2
 MOVE.L   #PAGE2,A3
 SUB.L    #3570,A3
STARLOP1:
 MOVE.L   #SORTDATA,A5
 MOVE.W   #15,D7
STARLOP2:
 MOVE.L   #0,D1
 MOVE.W   (A2),D1
 CMP.W    D1,D3
 BLT      STLO1
 SUB.W    #1,D4
STLO1:
 MOVE.L   A5,A6
 ASL.L    #1,D1
 ADD.L    D1,A6
 SUB.W    #1,(A6)
 ADD.L    #512,A2
 MOVE.L   #0,D1
 MOVE.W   (A3),D1
 CMP.W    D1,D3
 BLT      STLO2
 ADD.W    #1,D4
STLO2:
 MOVE.L   A5,A6
 ASL.L    #1,D1
 ADD.L    D1,A6
 ADD.W    #1,(A6)
 ADD.L    #512,A3
 SUB.W    #1,D7
 BNE      STARLOP2
 SUB.L    #7678,A2
 SUB.L    #7678,A3
PICKONE:
 CMP.W    #111,D4
 BEQ      FOUNDIT
 BGT      LOWER
 BRA      HIGHER
HIGHER:
 ADD.L    #2,A4
 ADD.W    #1,D3
 ADD.W    (A4),D4
 CMP.W    #111,D4
 BLT      HIGHER
 BRA      FOUNDIT
LOWER:
 SUB.W    (A4),D4
 SUB.L    #2,A4
 SUB.W    #1,D3
 CMP.W    #111,D4
 BGT      LOWER
 ADD.L    #2,A4
 ADD.W    #1,D3
 ADD.W    (A4),D4
FOUNDIT:
 MOVE.L   A4,A6
 SUB.L    A5,A6
 MOVE.L   A6,D1
 LSR.W    #1,D1
 MOVE.W   D1,(A1)+
```

```
   SUB.L     #1,D0
   BNE       STARLOP1
   MOVEM.L   (A7)+,D0-D7/A0-A6
   RTS
  *********
  AVERAGE:
   MOVEM.L   D0-D2/A0-A3,-(A7)
   MOVE.L    #PAGE2,A0
   SUB.L     #512,A0
   MOVE.L    #PAGE2,A1
   MOVE.L    #PAGE2,A2
   ADD.L     #512,A2
   MOVE.L    #PAGE1,A3
   MOVE.L    #$19300,D0
  BIGONE:
   CLR.L     D1
   CLR.L     D2
   SUB.L     #2,A0
   SUB.L     #2,A1
   SUB.L     #2,A2
   MOVE.W    (A0)+,D2
   ADD.L     D2,D1
   MOVE.W    (A1)+,D2
   ADD.L     D2,D1
   MOVE.W    (A2)+,D2
   ADD.L     D2,D1
   MOVE.W    (A0)+,D2
   ADD.L     D2,D1
   MOVE.W    (A1)+,D2
   ADD.L     D2,D1
   MOVE.W    (A2)+,D2
   ADD.L     D2,D1
   MOVE.W    (A0),D2
   ADD.L     D2,D1
   MOVE.W    (A1),D2
   ADD.L     D2,D1
   MOVE.W    (A2),D2
   ADD.L     D2,D1
   DIVU      #9,D1
   MOVE.W    D1,(A3)+
   SUB.L     #1,D0
   BNE       BIGONE
   MOVEM.L   (A7)+,D0-D2/A0-A3
   RTS
  *********
  HIGHPASS:
   MOVEM.L   D0-D5/A0-A4,-(A7)
   MOVE.L    #PAGE1,A1
   MOVE.L    #PAGE2,A2
   MOVE.L    #$19300,D0
   CLR.L     D1
   CLR.L     D2
   CLR.L     D3
   CLR.L     D4
   SUB.L     #1028,A2
   MOVE.W    #5,D1
  PASINIT1:
   MOVE.W    #5,D2
  PASINIT2:
   MOVE.W    (A2)+,D4
   ADD.L     D4,D3
   SUB.W     #1,D2
   BNE       PASINIT2
   ADD.L     #502,A2
   SUB.L     #1,D1
   BNE       PASINIT1
   MOVE.L    #PAGE2,A0
   SUB.L     #1028,A0
   MOVE.L    #PAGE2,A1
   SUB.L     #1020,A1
   MOVE.L    #PAGE2,A2
   MOVE.L    #PAGE1,A3
  PASSLOP1:
   MOVE.W    #5,D1
   ADD.L     #2,A1
  PASSLOP2:
   CLR.L     D4
   MOVE.W    (A1),D4
   ADD.L     D4,D3
   MOVE.W    (A0),D4
   SUB.L     D4,D3
   ADD.L     #512,A0
   ADD.L     #512,A1
```

continued

```
 SUB.L    #1,D1
 BNE      PASSLOP2
 ADD.L    #2,A0
 SUB.L    #2560,A0
 SUB.L    #2560,A1
 MOVE.L   D3,D4
 DIVU     #25,D4
 MOVE.W   (A2)+,D5
 LSL.W    #1,D5 *
* ADD.W    #2047,D5
 SUB.W    D4,D5
 CMP.W    #0,D5
 BGT      TOBIG
 MOVE.W   #0,D5
TOBIG:
 CMP.W    #4095,D5
 BLT      RIGHTON
 MOVE.W   #4095,D5
RIGHTON:
 MOVE.W   D5,(A3)+
 SUB.L    #1,D0
 BNE      PASSLOP1
 MOVEM.L  (A7)+,D0-D5/A0-A4
 RTS
*********
NEGATIVE:
 MOVEM.L  D0-D3/A0-A2,-(A7)
 MOVE.L   #PAGE1,A1
 MOVE.L   #PAGE2,A2
 MOVE.L   #$19300,D0
NEGAT0:
 MOVE.W   #4095,D3
 MOVE.W   (A2)+,D2
 SUB.W    D2,D3
 MOVE.W   D3,(A1)+
 SUB.L    #1,D0
 BNE      NEGAT0
 MOVEM.L  (A7)+,D0-D3/A0-A2
 RTS
*********
SHOW:
 MOVEM.L   D0-D3/A0,-(A7)
RESIN:
 CLR.L    D1
 MOVE.W   #4,D1
 MOVE.L   #$19300,D2
 MOVE.L   #PAGE1,A0
 MOVE.B   PAGE,D3
 CMP.B    #2,D3
 BNE      RESDO
 MOVE.L   #PAGE2,A0
RESDO:
 MOVE.W   (A0),D3
 ROR.W    D1,D3
 MOVE.W   D3,(A0)+
 SUB.L    #1,D2
 BNE      RESDO
 JSR      HISTGRAM
 MOVE.B   #9,BITPLANE
NEXTPLAN:
 CLR.L    D0
 MOVE.B   BITPLANE,D0
 SUB.B    #1,D0
 BEQ      STOP
 MOVE.B   D0,BITPLANE
 MOVE.W   #BASEROW,Y      *SELECT FIRST LINE
NEXTLINE:
 JSR      M               *POSITION POINTER TO START OF LINE
 JSR      WR              *SEND IT TO THE VECTRIX
 CLR.L    D0              *CLEAR FOR COMPARISON
 MOVE.W   Y,D0            *GET SCREEN Y VALUE
 ADD.W    #1,D0           *INCREMENT Y VALUE
 CMP.W    #TOPROW,D0      *HAVE WE DONE 441-38 = 403 LINES?
 BEQ      NEXTPLAN        *IF SO, THEN DO NEXT BIT PLANE
 MOVE.W   D0,Y            *IF NOT, THEN SET UP Y AND
 BRA      NEXTLINE        *DO NEXT LINE
STOP:
 MOVE.L   #$19300,D2
 SUB.L    #$32600,A0
RESUNDO:
 MOVE.W   (A0),D3
 ROL.W    D1,D3
 MOVE.W   D3,(A0)+
 SUB.L    #1,D2
 BNE      RESUNDO
 MOVEM.L   (A7)+,D0-D3/A0
```

```
 RTS
*********
BIGSHOW:
 MOVEM.L  D0-D3/A0,-(A7)
 CLR.L    D1
 MOVE.B   #$4,D1
 MOVE.L   #$19300,D2
 MOVE.L   #PAGE2,A0
BIGRESDO:
 MOVE.W   (A0),D3
 ROR.W    D1,D3
 MOVE.W   D3,(A0)+
 SUB.L    #1,D2
 BNE      BIGRESDO
 MOVE.B   #9,BITPLANE
BIGPLAN:
 CLR.L    D0
 MOVE.B   BITPLANE,D0
 SUB.B    #1,D0
 BEQ      STOP
 MOVE.B   D0,BITPLANE
 MOVE.W   #0,X
 MOVE.W   #0,Y
 MOVE.L   #24556,BIGX
BIGLINE:
 JSR      M
 JSR      BIGWR1
 CLR.L    D0
 MOVE.W   Y,D0
 ADD.W    #1,D0
 CMP.W    #479,D0
 BEQ      BIGPLAN
 MOVE.W   D0,Y
 JSR      M
 JSR      BIGWR2
 SUB.L    #2,BIGX
 CLR.L    D0
 MOVE.W   Y,D0
 ADD.W    #1,D0
 CMP.W    #479,D0
 BEQ      BIGPLAN
 MOVE.W   D0,Y
 BRA      BIGLINE
BIGSTOP:
 MOVE.L   #$19300,D2
 MOVE.L   #PAGE2,A0
BIGUNDO:
 MOVE.W   (A0),D3
 ROL.W    D1,D3
 MOVE.W   D3,(A0)+
 SUB.L    #1,D2
 BNE      BIGUNDO
 MOVEM.L   (A7)+,D0-D3/A0
 RTS
*********
*         HISTGRAM SUBROUTINE
*         MAKES A HISTOGRAM OF THE PAGE TO BE SHOWN
HISTGRAM:
 MOVEM.L  D0-D3/A0-A1,-(A7)
 MOVE.L   #HISTDATA,A1
 MOVE.W   #256,D0
HISTA0:
 MOVE.L   #0,(A1)+
 SUB.W    #1,D0
 BNE      HISTA0
 MOVE.L   #HISTDATA,A1
 MOVE.L   #$19300,D0
 MOVE.L   #PAGE1,A0
 MOVE.B   PAGE,D2
 CMP.B    #1,D2
 BEQ      HISTA1
 MOVE.L   #PAGE2,A0
HISTA1:
 CLR.L    D1
 MOVE.W   (A0)+,D1
 AND.W    #$00FF,D1
 LSL.W    #2,D1
 ADD.L    #1,0(A1,D1)
 SUB.L    #1,D0
 BNE      HISTA1
 CLR.W    D3
 MOVE.L   #255,D0
 MOVE.W   #BASECOL1,D1
 MOVE.B   PAGE,D2
```

continued

```
 CMP.B    #1,D2
 BEQ      HISTOUT
 MOVE.W   #BASECOL2,D1
HISTOUT:
 CLR.L    D2
 MOVE.L   (A1)+,D2
 ADD.L    #63,D2
 CMP.L    #$7FF,D2
 BLT      HISTA2
 MOVE.L   #$7FF,D2
HISTA2:
 LSR.W    #5,D2
 ADD.W    #4,D2
 JSR      TEST
 MOVE.B   #'C',VECOUT
 JSR      TEST
 MOVE.B   D3,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #'M',VECOUT
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROR.W    #8,D1
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROL.W    #8,D1
 JSR      TEST
 MOVE.B   #4,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #'L',VECOUT
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROR.W    #8,D1
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROL.W    #8,D1
 JSR      TEST
 MOVE.B   D2,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #'C',VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 JSR      TEST
 MOVE.B   #'L',VECOUT
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROR.W    #8,D1
 JSR      TEST
 MOVE.B   D1,VECOUT
 ROL.W    #8,D1
 JSR      TEST
 MOVE.B   #68,VECOUT
 JSR      TEST
 MOVE.B   #0,VECOUT
 ADD.W    #1,D1
 ADD.W    #1,D3
 SUB.L    #1,D0
 BPL      HISTOUT
 MOVEM.L  (A7)+,D0-D3/A0-A1
 RTS
*********
*         M SUBROUTINE
*         PERFORMS VECTRIX
                          M"
                            O"VE COMMAND
*         'X','Y' MUST CONTAIN THE SCREEN LOCATION YOU WISH TO MOVE TO
M:
 MOVEM.L   D0,-(A7)
 JSR      TEST
 MOVE.B   #'M',VECOUT
 MOVE.W   X,D0
 JSR      TEST
 MOVE.B   D0,VECOUT
 LSR      #8,D0
 JSR      TEST
 MOVE.B   D0,VECOUT
 MOVE.W   Y,D0
 JSR      TEST
 MOVE.B   D0,VECOUT
```

```
 LSR      #8,D0
 JSR      TEST
 MOVE.B   D0,VECOUT
 MOVEM.L   (A7)+,D0
 RTS
*************************************************************************
*         WR SUBROUTINE
*         WRITE GRAPHICS RAM TO SCREEN, ONE LINE AND ONE PLANE PER INVOCATION
*         REQUIRES 'X','Y' TO POINT TO SCREEN LOCATION OF 16 PIXEL BLOCK
*         AT WHICH LINE WILL BE DRAWN
*         REQUIRES 'BITPLANE' TO HOLD THE NUMBER OF THE BITPLANE WE WILL USE
*         'Y' IS USED AS AN OFFSET INTO THE GRAPHICS BUFFERS TO DETERMINE
*         PRESENT LINE
*
*     *********** VECTRIX MUST BE IN FLASH MODE************
*     ***************** FOR WR TO RUN **********************
*
WR:
 MOVEM.L  D0-D1/A0,-(A7)
 MOVE.W   Y,D0  *GET SCREEN Y VALUE
 SUB.W    #BASEROW,D0     *GRAPHICS Y = SCREEN Y - BASEROW
 MULU     #COLUMNS,D0     *POINT TO FIRST PIXEL IN LINE
 LSL.L    #1,D0           *TIMES TWO FOR 16 BIT DATA
 MOVE.L   D0,WRLINE       *SAVE THIS AS OFFSET TO PAGE
 MOVE.L   #COLUMNS,D0     *FIGURE HOW MANY 16 BIT BLOCKS PER LINE
 DIVU     #16,D0          *
 MOVE.B   D0,BLOCKS       *AND SAVE THE RESULT
 BSR      TEST            *CHECK FOR BUSY TRANSMIT REGISTER
 MOVE.B   #'W',VECOUT     *VECTRIX COMMAND =
                                        W"R BITPLANE COUNT


                                                          B"SR      TEST
 MOVE.B   #'R',VECOUT
 BSR      TEST
 MOVE.B   BITPLANE,VECOUT
 BSR      TEST
 MOVE.B   D0,VECOUT       *BLOCKS TO OUTPUT
 BSR      TEST
 MOVE.B   #0,VECOUT       *DONE WITH HEADER OF COMMAND, NOW FOR DATA
 MOVE.B   BITPLANE,D0     *BITPLANE (1..8) TO BIT (0..7)
 SUB.B    #1,D0           *BY SUBTRACTION OF ONE
 MOVE.B   D0,BIT          *AND SAVING THE RESULT
 MOVE.L   #PAGE1,A0       *POINT TO START OF PRIMARY PAGE AS CONTOUR
 MOVE.B   PAGE,D0         *GET PAGE NUMBER
 CMP      #2,D0 *AND SEE IF WE SHOULD USE SECONDARY PAGE
 BNE      NOTPAGE2        *IF PAGE # <> 2 THEN DON'T USE PAGE 2
 MOVE.L   #PAGE2,A0       *IF PAGE # = 2 THEN POINT TO SECONDARY
NOTPAGE2:
 ADDA.L   WRLINE,A0       *ADD OFFSET TO POINT TO A LINE WITHIN PAGE
WRLOOP1:
 MOVE.B   #16,BYTE        *NUMBER OF BYTES PER BLOCK
 CLR.L    D0
WRLOOP2:
 CLR.L    D1
 MOVE.B   BIT,D1          *USE D1 FOR BIT POINTER
 ADD.L    #1,A0
 BTST     D1,(A0)+        *SEE IF BIT IS ON
 BEQ      DONOTADD
 ADD.L    #$10000,D0      *IF BIT IS ON, REFLECT IN OUTPUT WORD
DONOTADD:
 LSR.L    #1,D0           *SHIFT INTO LOW BYTE
 SUB.B    #1,BYTE         *WE'VE NOW ONE LESS BYTE TO FINISH BLOCK
 BEQ      BLOCKSUB        *IF BLOCK FINISHED THEN SEND IT OUT
 BRA      WRLOOP2         *IF NOT, GET NEXT BIT
BLOCKSUB:
 BSR      TEST            *WE FINISHED OUTPUT WORD, NOW SEND IT
 MOVE.B   D0,VECOUT       *TO THE VECTRIX, LOW BYTE
 LSR.W    #8,D0           *SHIFT TO HIGH BYTE
 BSR      TEST
 MOVE.B   D0,VECOUT       *TO THE VECTRIX, HIGH BYTE
 SUB.B    #1,BLOCKS       *WE'VE NOW ONE LESS BLOCK TO FINISH LINE
 BNE      WRLOOP1         *IF LINE NOT DONE, GO DO NEXT BLOCK
 MOVEM.L  (A7)+,D0-D1/A0
WRDONE:
 RTS
*************************************************************************
*         BIGWR1 SUBROUTINE
*         WRITE FULLSCREEN PAGE 2
*         WRITE GRAPHICS RAM TO SCREEN, ONE LINE AND ONE PLANE PER INVOCATION
*         REQUIRES 'X','Y' TO POINT TO SCREEN LOCATION OF 16 PIXEL BLOCK
*         AT WHICH LINE WILL BE DRAWN
*         REQUIRES 'BITPLANE' TO HOLD THE NUMBER OF THE BITPLANE WE WILL USE
```

continued

```
*          'Y' IS USED AS AN OFFSET INTO THE GRAPHICS BUFFERS TO DETERMINE
*          PRESENT LINE
*
*     *********** VECTRIX MUST BE IN FLASH MODE************
*     ***************** FOR WR TO RUN **********************
*
BIGWR1:
 MOVEM.L  D0-D2/A0,-(A7)
 MOVE.W   0,D0
 MOVE.B   #42,BLOCKS
 BSR      TEST
 MOVE.B   #'W',VECOUT
 BSR      TEST
 MOVE.B   #'R',VECOUT
 BSR      TEST
 MOVE.B   BITPLANE,VECOUT
 BSR      TEST
 MOVE.B   #42,VECOUT
 BSR      TEST
 MOVE.B   #0,VECOUT
 MOVE.B   BITPLANE,D0
 SUB.B    #1,D0
 MOVE.B   D0,BIT
 MOVE.L   #PAGE2,A0
 ADDA.L   BIGX,A0
BIGLOOP1:
 MOVE.B   #8,BYTE
 CLR.L    D0
BIGLOOP2:
 CLR.L    D1
 MOVE.B   BIT,D1
 ADD.L    #512,A0
 MOVE.W   (A0),D2
 SUB.L    #512,A0
 ADD.W    (A0),D2
 ADD.L    #2,A0
 ADD.W    (A0),D2
 ADD.L    #512,A0
 ADD.W    (A0),D2
 SUB.L    #2,A0
 LSR.W    #2,D2
 BTST     D1,D2
 BEQ      BNOT0
 ADD.L    #$10000,D0
BNOT0:
 LSR.L    #1,D0
 MOVE.W   (A0),D2
 ADD.L    #512,A0
 ADD.W    (A0),D2
 ADD.L    #2,A0
 ADD.W    (A0),D2
 SUB.L    #512,A0
 ADD.W    (A0),D2
 SUB.L    #2,A0
 LSR.W    #2,D2
 BTST     D1,D2
 BEQ      BIGNOT
 ADD.L    #$10000,D0
BIGNOT:
 LSR.L    #1,D0
 SUB.B    #1,BYTE
 BEQ      BIGSUB
 BRA      BIGLOOP2
BIGSUB:
 BSR      TEST
 MOVE.B   D0,VECOUT
 LSR.W    #8,D0
 BSR      TEST
 MOVE.B   D0,VECOUT
 SUB.B    #1,BLOCKS
 BNE      BIGLOOP1
 MOVEM.L  (A7)+,D0-D2/A0
 RTS
*********
*          BIGWR2 SUBROUTINE
*     *********** VECTRIX MUST BE IN FLASH MODE************
*     ***************** FOR WR TO RUN **********************
*
BIGWR2:
 MOVEM.L  D0-D2/A0,-(A7)
 MOVE.W   0,D0
 MOVE.B   #42,BLOCKS
 BSR      TEST
 MOVE.B   #'W',VECOUT
 BSR      TEST
 MOVE.B   #'R',VECOUT
```

```
 BSR       TEST
 MOVE.B    BITPLANE,VECOUT
 BSR       TEST
 MOVE.B    #42,VECOUT
 BSR       TEST
 MOVE.B    #0,VECOUT
 MOVE.B    BITPLANE,D0
 SUB.B     #1,D0
 MOVE.B    D0,BIT
 MOVE.L    #PAGE2,A0
 ADDA.L    BIGX,A0
BIGLOO21:
 MOVE.B    #8,BYTE
 CLR.L     D0
BIGLOO22:
 CLR.L     D1
 MOVE.B    BIT,D1
 ADD.L     #512,A0
 MOVE.W    (A0),D2
 SUB.L     #2,A0
 ADD.W     (A0),D2
 SUB.L     #512,A0
 ADD.W     (A0),D2
 ADD.L     #2,A0
 ADD.W     (A0),D2
 ADD.L     #512,A0
 LSR.W     #2,D2
 BTST      D1,D2
 BEQ       BNOT20
 ADD.L     #$10000,D0
BNOT20:
 LSR.L     #1,D0
 MOVE.W    (A0),D2
 BTST      D1,D2
 BEQ       BIGNOT2
 ADD.L     #$10000,D0
BIGNOT2:
 LSR.L     #1,D0
 SUB.B     #1,BYTE
 BEQ       BIGSUB2
 BRA       BIGLOO22
BIGSUB2:
 BSR       TEST
 MOVE.B    D0,VECOUT
 LSR.W     #8,D0
 BSR       TEST
 MOVE.B    D0,VECOUT
 SUB.B     #1,BLOCKS
 BNE       BIGLOO21
 MOVEM.L   (A7)+,D0-D2/A0
 RTS
*********
*          TEST SUBROUTINE
*          WAITS UNTIL THE TRANSMIT BUFFER IS EMPTY
TEST:
 BTST      #0,VECSTAT
 BEQ       TEST
 BTST      #7,VECSTAT
 BEQ       TEST
 RTS
*********
*          SUBROUTINE BIAS
*          MOVES PRIMARY GRAPHICS DATA TO SECONDARY DATA SPACE
BIAS:
 MOVEM.L   D0-D4/A0-A2,-(A7)
 MOVE.L    #$19300,D0
 MOVE.L    #PAGE1,A0
 MOVE.L    #PAGE2,A1
 MOVE.L    A1,D1
 ADD.L     #514,D1
 MOVE.L    D1,A2
 CLR.L     D2
BIASLOOP:
 MOVE.W    (A1)+,D2
 SUB.W     (A2)+,D2
 ASR.W     #1,D2
 ADD.W     #2047,D2
 MOVE.W    D2,(A0)+
 SUB.L     #1,D0
 BNE       BIASLOOP
 MOVEM.L   (A7)+,D0-D4/A0-A2
 RTS
```

continued

```
*********
*         SUBROUTINE PTOS
*         MOVES PRIMARY GRAPHICS DATA TO SECONDARY DATA SPACE
PTOS:
 MOVEM.L  D0/A0-A1,-(A7)
 MOVE.L   #$19300,D0
 MOVE.L   #PAGE1,A0
 MOVE.L   #PAGE2,A1
PTOSLOOP:
 MOVE.W   (A0)+,(A1)+
 SUB.L    #1,D0
 BNE      PTOSLOOP
 MOVEM.L  (A7)+,D0/A0-A1
 RTS
*********
*         SUBSTOP SUBROUTINE
*         SUBTRACTS SECONDARY FROM PRIMARY, RESULTS TO PRIMARY
SUBSTOP:
 MOVEM.L  D0-D2/A0-A1,-(A7)
 MOVE.L   #$19300,D0
 MOVE.L   #PAGE1,A0
 MOVE.L   #PAGE2,A1
SUBLOOP:
 CLR.L    D1
 CLR.L    D2
 MOVE.W   (A1)+,D1
 MOVE.W   (A0),D2
 SUB.L    D1,D2
 BGE      SUBNOT
 CLR.L    D2
SUBNOT:
 MOVE.W   D2,(A0)+
 SUB.L    #1,D0
 BNE      SUBLOOP
 MOVEM.L  (A7)+,D0-D2/A0-A1
 RTS
*********
*         SUBROUTINE ACSYNC
*         SYNCRONIZES ROW READS WITH 60Hz LINE FREQUENCY
ACSYNC:
 MOVEM.L  D0-D1/A0,-(A7)
 MOVE.W   #0,D0
SYNC0:
 MOVE.W   #$C700,A_D_DATA
SYNC1:
 MOVE.W   A_D_DATA,D1
 BTST     #7,D1
 BEQ      SYNC1
 ROL.W    #8,D1
 BCHG     #11,D1
 NOT.W    D1
 AND.W    #$0FFF,D1
 CMP.W    #5,D0
 BGT      SYNC2
 CMP.W    #200,D1
 BLT      SYNC0
 ADD.W    #1,D0
 BRA      SYNC0
SYNC2:
 CMP.W    #272,D1
 BGT      SYNC0
 MOVEM.L  (A7)+,D0-D1/A0
 RTS
*********
*         SUBROUTINE RCAIN
*         READS A FRAME FROM RCA SID-504 CCD
RCAIN:
 MOVEM.L  D0-D7/A0-A1,-(A7)
 MOVE.B   #$4,SELECT      *SELECT PARALLEL PORT (CCD)
 MOVE.L   #PAGE1,A0
 MOVE.L   A0,A1
 ADD.L    #806,A1
 JSR      EXPTIME
 JSR      CV
 MOVE.W   #256,D0
VERTCLK:
 MOVE.B   HORTIME,D2
 JSR      ACSYNC
VERT0:
 SUB.B    #1,D2
 BNE      VERT0
 MOVE.B   #$20,PARDATA
 MOVE.B   #$60,PARDATA
 MOVE.B   VERTIME,D2
VERT2:
 SUB.B    #1,D2
```

```
   BNE       VERT2
   JSR       HORIZCLK
   SUB.W     #1,D0
   BNE       VERTCLK
   MOVEM.L   (A7)+,D0-D7/A0-A1
   JSR       FLIP
   JSR       MANSCR
   RTS
  *********
  *          SUBROUTINE HORIZCLK
  HORIZCLK:
   MOVEM.L   D0-D2,-(A7)
   MOVE.W    #14,D0
  DREGS1:
   MOVE.B    #$40,PARDATA
   MOVE.B    #$60,PARDATA
   MOVE.B    HORTIME,D2
  HZL9:
   SUB.B     #1,D2
   BNE       HZL9
   SUB.W     #1,D0
   BNE       DREGS1
   MOVE.W    #403,D0
  HORIZ1:
   MOVE.B    #$40,PARDATA
   MOVE.B    #$60,PARDATA
   MOVE.B    ADTIME,D2
  HZL0:
   SUB.B     #1,D2
   BNE       HZL0
   MOVE.W    #$C100,A_D_DATA
  ADDONE:
   MOVE.W    A_D_DATA,D1
   BTST      #7,D1
   BEQ       ADDONE
   ROL.W     #8,D1
   BCHG      #11,D1
   NOP
   AND.W     #$0FFF,D1
   MOVE.W    D1,(A0)+
   SUB.W     #1,D0
   BNE       HORIZ1
   MOVE.W    #14,D0
  DREGS:
   MOVE.B    #$40,PARDATA
   MOVE.B    #$60,PARDATA
   MOVE.B    HORTIME,D1
  DREGO:
   SUB.B     #1,D1
   BNE       DREGO
   SUB.W     #1,D0
   BNE       DREGS
   MOVEM.L   (A7)+,D0-D2
   RTS
  *********
  *          SUBROUTINE CLEAR CCD
  CLEARCCD:
   MOVEM.L   D0-D2,-(A7)
   MOVE.B    #$4,SELECT      *SELECT PARALLEL PORT (CCD)
   MOVE.W    #256,D0
  CVERTCLK:
   MOVE.B    #1,D2
  CVERT0:
   SUB.B     #1,D2
   BNE       CVERT0
   MOVE.B    #$20,PARDATA
   MOVE.B    #$60,PARDATA
   MOVE.B    VERTIME,D2
  CVERT2:
   SUB.B     #1,D2
   BNE       CVERT2
   JSR       CHORIZ
   SUB.W     #1,D0
   BNE       CVERTCLK
   JSR       CV
   MOVEM.L   (A7)+,D0-D2
   RTS
  *********
  *          SUBROUTINE CHORIZ
  CHORIZ:
   MOVEM.L   D0-D1,-(A7)
   MOVE.W    #427,D0
```

continued

```
CHORIZ1:
 MOVE.B   #$40,PARDATA
 MOVE.B   #$60,PARDATA
 MOVE.B   HORTIME,D1
CXHZ0:
 SUB.B    #1,D1
 BNE      CXHZ0
 SUB.W    #1,D0
 BNE      CHORIZ1
 MOVEM.L  (A7)+,D0-D1
 RTS
*********
*         SUBROUTINE CV MOVE FROM IMAGE TO STORAGE
CV:
 MOVEM.L  D0-D3,-(A7)
 MOVE.W   #256,D2
CV1:
 MOVE.B   #$20,PARDATA
 MOVE.B   #$60,PARDATA
 MOVE.B   VERTIME,D1
CV2:
 SUB.B    #1,D1
 BNE      CV2
 MOVE.W   #3,D3
CVHOR:
 MOVE.B   #$40,PARDATA
 MOVE.B   #$60,PARDATA
 MOVE.B   HORTIME,D0
CV0:
 SUB.B    #1,D0
 BNE      CV0
 SUB.W    #1,D3
 BNE      CVHOR
 SUB.W    #1,D2
 BNE      CV1
 MOVEM.L  (A7)+,D0-D3
 RTS
*********
*         SUBROUTINE EXPTIME
*         TIMES OUT CCD EXPOSURE
EXPTIME:
 MOVEM.L  D0-D1/A1,-(A7)
 MOVE.B   $1A,SYSIO
 MOVE.L   #EXPMSG,A1
 JSR      MSGOUT
 JSR      TESTOUT
 MOVE.B   EXPTIME1,D1
 MOVE.B   D1,SYSIO
 JSR      TESTOUT
 MOVE.B   EXPTIME2,D1
 MOVE.B   D1,SYSIO
 JSR      TESTOUT
 MOVE.B   EXPTIME3,D1
 MOVE.B   D1,SYSIO
 JSR      TESTOUT
 MOVE.B   #'.',D1
 MOVE.B   D1,SYSIO
 JSR      TESTOUT
 MOVE.B   EXPTIME4,D1
 MOVE.B   D1,SYSIO
 JSR      CLEARCCD
 JSR      CLEARCCD
 JSR      CLEARCCD
 MOVE.W   EXPOSURE,D1      *  556701 = 1 SECOND
BIGTIME:
 MOVE.L   #55670,D0
TIMEOUT:
 SUB.L    #1,D0
 BNE      TIMEOUT
 SUB.W    #1,D1
 BNE      BIGTIME
 MOVEM.L  (A7)+,D0-D1/A1
 RTS

*********
*         SUBROUTINE FLIP
*         FLIPS IMAGE TO FIT SCREEN
FLIP:
 MOVEM.L  D0-D2/A0-A1,-(A7)
 MOVE.L   #$19300,D0
 MOVE.L   #PAGE2,A1
FLIP0:
 MOVE.W   #0,(A1)+
 SUB.L    #1,D0
 BNE      FLIP0
 MOVE.L   #PAGE2,A1
```

```
 MOVE.W   #0,D1
 MOVE.L   #PAGE1,A0
FLIP1:
 MOVE.W   #403,D0
FLIP2:
 MOVE.W   (A0)+,(A1)
 ADD.L    #512,A1
 SUB.W    #1,D0
 BNE      FLIP2
 MOVE.L   #PAGE2,A1
 ADD.W    #2,D1
 CMP.W    #512,D1
 BEQ      FLIPOUT
 ADD.W    D1,A1
 BRA      FLIP1
FLIPOUT:
 MOVE.L   #PAGE1,A0
 MOVE.L   #PAGE2,A1
 MOVE.L   #$19300,D0
FLIP3:
 MOVE.W   (A1)+,(A0)+
 SUB.L    #1,D0
 BNE      FLIP3
 MOVEM.L  (A7)+,D0-D2/A0-A1
 RTS
*********
*         VECINIT SUBROUTINE
*         INITIALIZE VECTRIX FOR 2D, BLACK SCREEN, REPLACE MODE,
*         HEX MODE OPERATION
VECINIT:
 MOVEM.L  D0/A0,-(A7)
 MOVE.B   #$4,SELECT      *SELECT VECTRIX
 MOVEA.L  #VECDATA,A0
LOOP1:
 MOVE.B   (A0)+,D0
 CMP.B    #'+',D0         *EQUAL TO END CHARACTER?
 BEQ      INITDONE
 BSR      TEST
 MOVE.B   D0,VECOUT
 BRA      LOOP1
INITDONE:
 MOVEM.L  (A7)+,D0/A0
 RTS
*********
*         TESTIN SUBROUTINE
*         WAITS UNTIL THE TRANSMIT BUFFER IS EMPTY AND THE RECIEVE BUFFER
*         CONTAINS AN INPUT CHARACTER
TESTIN:
 JSR      TIME
 JSR      TESTOUT
 BTST     #1,SYSST
 BEQ      TESTIN
 RTS
*********
*         INNOTIME SUBROUTINE
*         SAME AS TESTIN, BUT NO TIME
INNOTIME:
 JSR      TESTOUT
 BTST     #1,SYSST
 BEQ      INNOTIME
 RTS
*********
*         TEST SUBROUTINE
*         WAITS UNTIL THE TRANSMIT BUFFER IS EMPTY
TESTOUT:
 BTST     #0,SYSST
 BEQ      TESTOUT
 RTS
*********
*         REPORT PROGRAM
*         GENERATES OUTPUT LISTING OF VITAL STATS OF GROUPS
REPORT:
 MOVEM.L  D0-D5/A0-A1,-(A7)
 MOVE.L   #RPTMSG0,A1
 JSR      MSGOUT
 MOVE.L   #256,D3
 MOVE.L   #AREA,A0
REPLP:
 SUB.L    #1,D3
 BEQ      REPDONE
 MOVE.L   D3,D1
 MULU     #4,D1
 MOVE.L   0(A0,D1),D2
 BEQ      REPLP
```

continued

```
  MOVE.L   D3,D0
  MOVE.L   #RPTMSG1,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  MOVE.L   D2,D0
  MOVE.L   #RPTMSG2,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  MOVE.L   #BRIGHT,A0
  MOVE.L   0(A0,D1),D0
  MOVE.L   #RPTMSG3,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  DIVU     #2,D1
  MOVE.L   #VMAX,A0
  CLR.L    D0
  MOVE.W   0(A0,D1),D0
  MOVE.L   #RPTMSG4,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  MOVE.L   #VMIN,A0
  CLR.L    D0
  MOVE.W   0(A0,D1),D0
  MOVE.L   #RPTMSG5,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  MOVE.L   #HMAX,A0
  CLR.L    D0
  MOVE.W   0(A0,D1),D0
  MOVE.L   #RPTMSG6,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  MOVE.L   #HMIN,A0
  CLR.L    D0
  MOVE.W   0(A0,D1),D0
  MOVE.L   #RPTMSG7,A1
  JSR      MSGOUT
  JSR      STASHD0
  JSR      INTOUT
  MOVE.L   #RPTMSG8,A1
  JSR      MSGOUT
  MOVE.L   #RPTMSG9,A1
  JSR      MSGOUT
  MOVE.L   #RPTMSGA,A1
  JSR      MSGOUT
  MOVE.L   #AREA,A0
  JSR      WAIT
  BRA      REPLP
 REPDONE:
  MOVEM.L  (A7)+,D0-D5/A0-A1
  RTS
 *********
 MSGOUT:
  MOVE.L   D0,-(A7)
 MSGLOOP:
  CLR.W    D0
  MOVE.B   (A1)+,D0
  CMP.B    #'+',D0
  BEQ      MSGOVER
  CMP.B    #'_',D0
  BNE      NOTCAR
  MOVE.B   #13,D0
  JSR      TESTOUT
  MOVE.B   D0,SYSIO
  MOVE.B   #10,D0
 NOTCAR:
  JSR      TESTOUT
  MOVE.B   D0,SYSIO
  BRA      MSGLOOP
 MSGOVER:
   MOVEM.L  (A7)+,D0
  RTS
 *********
 STASHD0:
 MOVEM.L  D1-D3/A0-A2,-(A7)
 MOVE.L   D0,D2
 MOVE.L   #MARK1,A0
 MOVE.B   #10,D3
LOOP:
 MOVE.L   D2,D1
```

```
 MOVE.L   D1,D0
 JSR      STUFFD0
 MOVE.L   #10,D0
 JSR      STUFFD0
 MOVE.B   #$2F,MATHCMD
 JSR      YANKD0
 MOVE.L   D0,D2
 JSR      STUFFD0
 MOVE.L   #10,D0
 JSR      STUFFD0
 MOVE.B   #$2E,MATHCMD
 JSR      YANKD0
 SUB.L    D0,D1
 ADD.B    #$30,D1
 MOVE.B   D1,-(A0)
 SUB.B    #1,D3
 BNE      LOOP
 MOVEM.L  (A7)+,D1-D3/A0-A2
 RTS
*********
YANKD0:
 MOVE.B   MATHDATA,D0
 ASL.L    #8,D0
 MOVE.B   MATHDATA,D0
 ASL.L    #8,D0
 MOVE.B   MATHDATA,D0
 ASL.L    #8,D0
 MOVE.B   MATHDATA,D0
 RTS
*********
STUFFD0:
 MOVE.B   D0,MATHDATA
 ROR.L    #8,D0
 MOVE.B   D0,MATHDATA
 ROR.L    #8,D0
 MOVE.B   D0,MATHDATA
 ROR.L    #8,D0
 MOVE.B   D0,MATHDATA
 RTS
*********
*         DECOUT SUBROUTINE
*         OUTPUTS TEN DIGIT NUMBER WITH DECIMAL POINT AFTER 3rd DIGIT
DECOUT:
 MOVEM.L  D0-D1/A0,-(A7)
 JSR      DECPOINT
 MOVE.B   #11,D1
 MOVE.L   #DECIMAL,A0
DECLOOP:
 MOVE.B   (A0)+,D0
 JSR      TESTOUT
 MOVE.B   D0,SYSIO
 SUB.B    #1,D1
 BNE      DECLOOP
 MOVEM.L  (A7)+,D0-D1/A0
 RTS
*********
*         INTOUT SUBROUTINE
*         OUTPUTS TEN DIGIT INTEGER REPRESENTATION OF LONG WORD
INTOUT:
 MOVEM.L  D0-D1/A0-A1,-(A7)
 JSR      KRUNCH
 MOVE.B   #10,D1
 MOVE.L   #DECIMAL,A0
INTLOOP:
 MOVE.B   (A0)+,D0
 CMP.B    #' ',D0
 BEQ      ZOUT
 JSR      TESTOUT
 MOVE.B   D0,SYSIO
ZOUT:
 SUB.B    #1,D1
 BNE      INTLOOP
 MOVEM.L  (A7)+,D0-D1/A0-A1
 RTS
*********
*         PLACES DECIMAL POINT IN DECIMAL NUMBER
DECPOINT:
 MOVEM.L  D0/A0,-(A7)
 MOVE.L   #DECIMAL,A0
 MOVE.B   #8,D0
 ADDA.L   #11,A0
FIXLOOP1:
 MOVE.B   -(A0),1(A0)
 SUB.B    #1,D0
```

continued

```
 BNE       FIXLOOP1
 MOVE.B    #'.',(A0)
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
*          KRUNCH SUBROUTINE
*          REMOVES UNWANTED ZEROS FROM NUMBERS
KRUNCH:
 MOVEM.L   D0-D1/A0,-(A7)
 MOVE.L    #DECIMAL,A0
 MOVE.B    #9,D0
KRLOOP:
 MOVE.B    (A0),D1
 CMP.B     #'0',D1
 BNE       KRDONE
 MOVE.B    #' ',(A0)+
 SUB.B     #1,D0
 BNE       KRLOOP
KRDONE:
 MOVEM.L   (A7)+,D0-D1/A0
 RTS
*********
*          WAIT SUBROUTINE
WAIT:
 MOVE.L    D0,-(A7)
 MOVE.L    #$2FFFFF,D0
WAITLOOP:
 SUB.L     #1,D0
 BNE       WAITLOOP
 MOVE.L    (A7)+,D0
 RTS
*********
*          SCREEN SUBROUTINE
*          SHOWS TOP MENU
SCREEN:
 MOVEM.L   D0/A0,-(A7)
 MOVE.L    #MENU1,A0
MESSAGE:
  MOVE.B    (A0)+,D0
 CMP.B     #'+',D0
 BEQ       MSGDONE
 CMP.B     #'_',D0
 BNE       NOCAR
 JSR       TESTOUT
 MOVE.B    #10,SYSIO
 MOVE.B    #13,D0
NOCAR:
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 BRA       MESSAGE
MSGDONE:
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
*          MANSCR SUBROUTINE
*          SHOWS MENU FOR MANUAL MODE
MANSCR:
 MOVEM.L   D0/A0,-(A7)
 MOVE.L    #MENU2,A0
MANMESS:
 MOVE.B    (A0)+,D0
 CMP.B     #'+',D0
 BEQ       MMSGDONE
 CMP.B     #'_',D0
 BNE       MNOCAR
 JSR       TESTOUT
 MOVE.B    #10,SYSIO
 MOVE.B    #13,D0
MNOCAR:
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 BRA       MANMESS
MMSGDONE:
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
*          AUTOSCR SUBROUTINE
*          SHOWS MENU FOR AUTOMATIC MODE
AUTOSCR:
 MOVEM.L   D0/A0,-(A7)
 MOVE.L    #MENU3,A0
AUTOMESS:
 MOVE.B    (A0)+,D0
 CMP.B     #'+',D0
 BEQ       AMSGDONE
 CMP.B     #' ',D0
```

```
 BNE       ANOCAR
 JSR       TESTOUT
 MOVE.B    #10,SYSIO
 MOVE.B    #13,D0
ANOCAR:
 JSR       TESTOUT
 MOVE.B    D0,SYSIO
 BRA       AUTOMESS
AMSGDONE:
 MOVEM.L   (A7)+,D0/A0
 RTS
*********
*          TIME SUBROUTINE
*          PUTS TIME AND DATE STUFF ON SCREEN
TIME:
 MOVEM.L   D0-D1/A0-A1,-(A7)
 JSR       CLOCK
 MOVE.L    #TIMEDATA,A0
 MOVE.L    #TIMEMSG1,A1
 JSR       MSGOUT
 JSR       TESTOUT
 MOVE.B    2(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    3(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    #'/',SYSIO
 JSR       TESTOUT
 MOVE.B    4(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    5(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    #'/',SYSIO
 JSR       TESTOUT
 MOVE.B    0(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    1(A0),SYSIO
 MOVE.L    #TIMEMSG2,A1
 JSR       MSGOUT
 JSR       TESTOUT
 MOVE.B    7(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    8(A0),SYSIO
 MOVE.L    #TIMEMSG3,A1
 JSR       MSGOUT
 JSR       TESTOUT
 MOVE.B    9(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    10(A0),SYSIO
 MOVE.L    #TIMEMSG4,A1
 JSR       MSGOUT
 JSR       TESTOUT
 MOVE.B    11(A0),SYSIO
 JSR       TESTOUT
 MOVE.B    12(A0),SYSIO
 MOVEM.L   (A7)+,D0-D1/A0-A1
 RTS
*********
*          CLOCK SUBROUTINE
*          GETS THE TIME AND STORES IT IN THE DATA AREA AS TIMEDATA
CLOCK:
 MOVEM.L   D0-D1/A0,-(A7)
 MOVE.L    #TIMEDATA,A0
 MOVE.B    #$C,D0
CLKLOOP1:
 MOVE.B    #$50,D1
 OR.B      D0,D1
 MOVE.B    D1,CLKCMD
 MOVE.B    CLKDATA,D1
 CMP.B     #5,D0
 BNE       CLKA1
 AND.B     #3,D1
CLKA1:
 ADD.B     #$30,D1
 MOVE.B    D1,(A0)+
 SUB.B     #1,D0
 BPL       CLKLOOP1
 MOVE.B    #0,CLKCMD
 MOVEM.L   (A7)+,D0-D1/A0
 RTS
*********
*          ACKMSG SUBROUTINE
*          ACKNOWLEGE A MENU CHOICE
ACKMSG:
 MOVEM.L   D0/A0,-(A7)
```

continued

```
 MOVE.L   A1,A0
 MOVE.L   #ACKMESS,A1
 JSR      MSGOUT
 MOVE.B   #22,D0
 ADDA.L   #2,A0 *DON'T PRINT CHARACTER OF CHOICE
ACKLOOP:
 JSR      TESTOUT
 MOVE.B   (A0)+,SYSIO
 SUB.B    #1,D0
 BNE      ACKLOOP
 MOVEM.L  (A7)+,D0/A0
 RTS
*********
 .DATA          *INITIALIZED DATA
VECDATA:  .DC.B     'G JM2 RE KF E0 C73 P4 63 69 63 473 320 473 320 69 '
 .DC.B     'P4 335 69 335 473 593 473 593 69 HX+'
MENU1:    .DC.B     $1A
 .DC.B     '          HOW FORTUNATE YOU ARE TO GAZE UPON THE'
 .DC.B     ' SPLENDOR OF_'
 .DC.B     '          *** THE TOTALY WOTALY FERRET CCD AND TELESCOPE'
 .DC.B     ' CONTROLLER PROGRAM ***_'
 .DC.B     ' by CLIFFORD HARRIS  of 99 MASON RD. YERINGTON, NV. 89447_'
 .DC.B     'DATE:  11/24/86_'
 .DC.B     '_'
 .DC.B     'COORDINATED UNIVERSAL TIME:     23h 59m 59s_'
 .DC.B     '__'
 .DC.B     '___'
 .DC.B     'PLEASE CHOOSE ONE OF THE FOLLOWING:_'
 .DC.B     '0 AUTOMATIC MODE_'
 .DC.B     '1 MANUAL MODE_'
 .DC.B     '2 SET TIMING  ( HORIZONTAL/2, VERTICAL/8, AD/8 )_'
 .DC.B     '3 SET EXPOSURE TIME IN 1/10 SECONDS (GIVE FOUR DIGITS)_'
 .DC.B     'Q QUIT_'
 .DC.B     '___+'
MENU2:    .DC.B     $1A
 .DC.B     '   THE WONDERFUL FERRET CCD AND TELESCOPE'
 .DC.B     ' CONTROLLER PROGRAM_'
 .DC.B     '   *#*#* THE INCREDIBLE MANUAL MODE'
 .DC.B     ' *#*#*_'
 .DC.B     ' '
 .DC.B     'DATE:  12/30/85_'
 .DC.B     '_'
 .DC.B     'COORDINATED UNIVERSAL TIME:     23h 59m 59s_'
 .DC.B     '__'
ACKMSG20: .DC.B     '0 CCD IN                '
ACKMSG21: .DC.B     '1 PAGE1 TO PAGE2        '
ACKMSG22: .DC.B     '2 AVERAGE              _'
ACKMSG23: .DC.B     '3 FIX BAD PIXELS        '
ACKMSG24: .DC.B     '4 COLORS               _'
ACKMSG25: .DC.B     '5 PAGE2 TO FULL SCREEN  '
ACKMSG26: .DC.B     '6 LOG                  _'
ACKMSG27: .DC.B     '7 DISPLAY PAGE1         '
ACKMSG28: .DC.B     '8 DISPLAY PAGE2        _'
ACKMSG29: .DC.B     '9 GROUPS                '
ACKMSG2A: .DC.B     'A MAGNIFY 2X           _'
ACKMSG2B: .DC.B     'B REPORT ON GROUPS      '
ACKMSG2C: .DC.B     'C SAVE PAGE1           _'
ACKMSG2D: .DC.B     'D SCREEN COLOR VALUES   '
ACKMSG2E: .DC.B     'E SWIRL COLORS         _'
ACKMSG2F: .DC.B     'F RESTORE               '
ACKMSG2G: .DC.B     'G DARK FIELD            '
ACKMSG2H: .DC.B     'H SUBTRACT PAGE1-PAGE2 _'
ACKMSG2I: .DC.B     'I FLAT FIELD            '
ACKMSG2J: .DC.B     'J BAS RELIEF            '
ACKMSG2K: .DC.B     'K CONTRAST ENHANCE     _'
ACKMSG2L: .DC.B     'L SCREEN CURSOR         '
ACKMSG2M: .DC.B     'M BLACK BACKGROUND      '
ACKMSG2N: .DC.B     'N NEGATIVE             _'
ACKMSG2O: .DC.B     'O ADD CONSTANT          '
ACKMSG2P: .DC.B     'P SUBTRACT CONSTANT     '
ACKMSG2R: .DC.B     'R PAGE1 TO DARKFIELD   _'
ACKMSG2S: .DC.B     'S PAGE1 TO FLATFIELD    '
ACKMSG2T: .DC.B     'T STAROUT 5 X 5         '
ACKMSG2U: .DC.B     'U 3-D GRAPH            _'
ACKMSG2V: .DC.B     'V HIGHPASS FILTER       '
ACKMSG2W: .DC.B     'W MAP BAD PIXELS        '
ACKMSG2X: .DC.B     'X STAROUT 15X15 MEDIAN _'
ACKMSG2Y: .DC.B     'Y MEAN AND STANDARD DEV '
ACKMSG2Z: .DC.B     'Z DIFFERENCE            '
          .DC.B     'Q QUIT         _'
 .DC.B     $0D,$0A,$0A,'+'
ACKMSG3C: .DC.B     '  SAVE IMAGE (0-9,A-Z) ?        '
ACKMSG3F: .DC.B     '  RESTORE IMAGE (0-9,A-Z) ?     '
RDYMSG2:  .DC.B     '  VOTT IS YOUR VISH?                _'
MENU3:    .DC.B     $1A
 .DC.B     '      THE MIND-EXPANDING FERRET CCD AND TELESCOPE'
```

```
 .DC.B     ' CONTROLLER PROGRAM_'
 .DC.B     '        !!!!  THE AMAZINGLY AMAZING AUTOMATIC'
 .DC.B     ' MODE !!!!__'
 .DC.B     'DATE:  12/30/85_'
 .DC.B     '_'
 .DC.B     'COORDINATED UNIVERSAL TIME:     23h 59m 59s_'
 .DC.B     '__'
 .DC.B     'PRESENT TELESCOPE COORDINATES:  R.A.  =  23h 59m 59.9s_'
 .DC.B     '     DEC.  = -89d 59m 59s_'
 .DC.B     '__'
 .DC.B     'INTEGRATION TIME:      99 min  59 sec'
 .DC.B     '_'
 .DC.B     'TIME LEFT ON THIS EXPOSURE:     99 min  59 sec'
 .DC.B     '__'
 .DC.B     'NOT A LOT TO CHOOSE FROM HERE, BUT IF YOU WISH TO STOP,'
 .DC.B     ' TYPE THE LETTER Q_'
 .DC.B     '_'
 .DC.B     'Q QUIT_'
 .DC.B     '+'
TIMEMSG1: .DC.B     $1B,'.0',$1B,'=#',$27,'+'
TIMEMSG2: .DC.B     $1B,'=%C+'
TIMEMSG3: .DC.B     $1B,'=%D+'
TIMEMSG4: .DC.B     $1B,'=%H+'
ACKMESS:  .DC.B     $1B,'=6 +'
ENDMSG:   .DC.B     '__',$1B,'.1_+'
RPTMSG0:  .DC.B     $1B,'=! '
 .DC.B     '** THE ASTOUNDING REPORT ON DIFFUSE OBJECTS **+'
RPTMSG1:  .DC.B     $1B,'=,!'
 .DC.B     'GROUP #  ',$1B,'=,*+'
RPTMSG2:  .DC.B     $1B,'=-!'
 .DC.B     'AREA  =    ',$1B,'=-2+'
RPTMSG3:  .DC.B     $1B,'=.!'
 .DC.B     'BRIGHTNESS      =    ',$1B,'=.2+'
RPTMSG4:  .DC.B     $1B,'=/!'
 .DC.B     'VERTICAL MAX    =    ',$1B,'=/2+'
RPTMSG5:  .DC.B     $1B,'=0!'
 .DC.B     'VERTICAL MIN    =    ',$1B,'=02+'
RPTMSG6:  .DC.B     $1B,'=1!'
 .DC.B     'HORIZONTAL MAX =    ',$1B,'=12+'
RPTMSG7:  .DC.B     $1B,'=2!'
 .DC.B     'HORIZONTAL MIN =    ',$1B,'=22+'
RPTMSG8:  .DC.B     $1B,'=3!'
 .DC.B     'R.A.  =    ',$1B,'=32+'
RPTMSG9:  .DC.B     $1B,'=4!'
 .DC.B     'DEC.  =    ',$1B,'=42+'
RPTMSGA:  .DC.B     $1B,'=5!'
 .DC.B     'MAGNITUDE       =    ',$1B,'=52+'
EXPMSG:   .DC.B     $1B,'=5!'
 .DC.B     'EXPOSURE TIME  =    ',$1B,'=52+'
ERRMSG:   .DC.B     $1B,'=6!'
 .DC.B     'ERROR, SO SORRY!    ',$1B,'=52+'
 .EVEN
FCB1:     .DC.B     3,'HALLAVGOPIC',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 .DC.B     0,0,0
FCB2:     .DC.B     2,'HELEN   PIC',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 .DC.B     0,0,0
DECIMAL:  .DC.B    '**********'
MARK1:    .DC.B    '***'     *MARKER FOR BACKWARDS FIXED POINTER
 .BSS
 .EVEN
EXPOSURE: .DS.W 1        *STORES CCD EXPOSURE IN SECONDS
LOGDATA:  .DS.W 4096
TIMEDATA: .DS.B $20
WCOLORS:  .DS.B $400
DMA:      .DS.B $80      *DMA BUFFER FOR DISK ACCESS
STAR_D0   .DS.L 1
CUSVALL:  .DS.W 1        *LEFT CURSOR HISTOGRAMS
CUSVALR:  .DS.W 1        *RIGHT CURSOR HISTOGRAMS
CUSVALH:  .DS.W 1        *HORIZONTAL VECTRIX CURSOR
CUSVALV:  .DS.W 1        *VERTICAL VECTRIX CURSOR
CUSVH:    .DS.W 1        *HORIZONTAL VECTRIX CURSOR
CUSVV:    .DS.W 1        *VERTICAL VECTRIX CURSOR
X:        .DS.W 1        *X,Y ARE SCREEN COORDINATES AT WHICH THE
Y:        .DS.W 1        *LINE WILL APPEAR
FRAMEX:   .DS.W 1
FRAMEY:   .DS.W 1
BIGX:     .DS.L 1
BIG:      .DS.W 1        *BIG TELLS SHOW IF BIG HAS BEEN SELECTED
BADMAP:   .DS.L $1000    *BAD PIXEL MAP
SORTDATA: .DS.W $1000    *STORAGE FOR STAROUT MEDIAN FILTER
VMAX:     .DS.W $1000    *MAXIMUM VERTICAL POSITIONS OF GROUPS
HMAX:     .DS.W $1000    *MAXIMUM HORIZONTAL POSITIONS OF GROUPS
VMIN:     .DS.W $1000    *MINIMUM VERTICAL POSITIONS OF GROUPS
HMIN:     .DS.W $1000    *MINIMUM HORIZONTAL POSITIONS OF GROUPS
```

continued

```
BRIGHT:    .DS.L $1000    *INTEGRATED BRIGHTNESS'S OF GROUPS
AREA:      .DS.L $1000    *INTEGRATED AREAS OF GROUPS
EXPTIME1:  .DS.B 1        *CCD EXPOSURE TIME
EXPTIME2:  .DS.B 1
EXPTIME3:  .DS.B 1
EXPTIME4:  .DS.B 1
HORTIME:   .DS.B 1
VERTIME:   .DS.B 1
ADTIME:    .DS.B 1
BIT:       .DS.B 1        *WILL NOT EXCEED 7 IF BITPLANE < 9
BITPLANE:  .DS.B 1        *MUST NOT EXCEED 8
BYTE:      .DS.B 1        *COUNTER FOR BYTES WITHIN THE BLOCK
BLOCKS:    .DS.B 1        *COUNTER FOR BLOCKS WITHIN A LINE
PAGE:      .DS.B 1        *POINTS AT WHICH PAGE TO USE
RED:       .DS.B 1
GREEN:     .DS.B 1
BLUE:      .DS.B 1
 .EVEN
HISTDATA:  .DS.L $100
WRLINE:    .DS.L 1        *PAGE BUFFER OFFSET POINTS TO LINE
STRBYTES:  .DS.B 256
STARDATA:  .DS.B $200     *STORAGE FOR SORTING ROUTINE
BLANK1:    .DS.B $400
PAGE1:     .DS.B $32600
BLANK2:    .DS.B $400
PAGE2:     .DS.B $32600
BLANK3:    .DS.B $400
PAGE3:     .DS.B $32600
BLANK4:    .DS.B $400
PAGE4:     .DS.B $32600
BLANK5:    .DS.B $400
 .END
```

NLP.C accompanies "Natural Language Processing in C" by Herbert Schildt, BYTE, December 1987, page 269.

```
/* NLP.C (4-8) */
/* Recursive descent NLP Example */
/* Be sure to end all input with a period */
#include "stdio.h"

#define MAX 100

#define NOUN 1
#define VERB 2
#define ADJ 3
#define ADV 4
#define DET 5
#define PREP 6
#define TERM 7

/* structure of the word database */
struct word {
  char word[20];
  char type;
};

struct word wdb[MAX];  /* array of db structures */

int db_pos=0; /* number of entries in wdb */

char s[80];    /* holds the sentence */
char *t_pos=0;   /* points into the sentence */
char token[80];  /* contains the word */

main()
{

  setup();

  printf("Enter Sentence: ");
  gets(s);
  t_pos=s;
  if(parse()) printf("Sentence OK\n");
  else printf("Error in sentence\n");
}

setup()
{
```

```
  assert_wdb("door",NOUN);
  assert_wdb("window",NOUN);
  assert_wdb("house",NOUN);
  assert_wdb("child",NOUN);
  assert_wdb("has",VERB);
  assert_wdb("runs",VERB);
  assert_wdb("plays",VERB);
  assert_wdb("large",ADJ);
  assert_wdb("quickly",ADV);
  assert_wdb("the",DET);
  assert_wdb("a",DET);
  assert_wdb("to",PREP);
  assert_wdb(".",TERM);
}

/* place facts into database */
assert_wdb(word,type)
char *word;
int type;
{

  if(db_pos<MAX) {
    strcpy(wdb[db_pos].word,word);
    wdb[db_pos].type=type;
    db_pos++;
  }
  else printf("Word database full.\n");
}

/* Context-free recursive descent NLP parser */
parse()
{

  if(!nounphrase()) return 0;
  if(!verbphrase()) return 0;
  if(!terminator()) return 0;
  return 1;

}

/* read a noun phrase from the input stream */
nounphrase()
{
  char type;

  get_token();

  type=find_type(token);
  switch(type) {
    case DET:
      get_token();
      type=find_type(token);
      if(type==NOUN) return 1;
      else if(type==ADJ) {
        get_token();
        type=find_type(token);
        if(type==NOUN) return 1;
      }
      break;
    case PREP:
      return nounphrase();
  }
  return 0;
}

/* read a verb phrase */
verbphrase()
{
  char type,*pos;

  get_token();

  type=find_type(token);
  if(type!=VERB) return 0;  /* must start with a verb */
  pos=t_pos; /* save current position for backtracking */

  /* verb + adverb + NP */
  if(verb_adv_np()) return 1;

  /* verb +NP */
  t_pos=pos;  /* back up */
  if(verb_np()) return 1;
```

continued

```
  /* verb+adverb -- no NP */
  t_pos=pos;
  if(verb_adv()) return 1;

  /* just verb */
  return 1;  /* error in verb phrase */
}

verb_np()
{
  /* verb + NP */
  return nounphrase();
}

verb_adv_np()
{
  char type;

  get_token();
  type=find_type(token);

  if(type==ADV && nounphrase()) return 1;
  return 0;
}

verb_adv()
{
  char type;

  get_token();
  type=find_type(token);

  return (type==ADV);
}

terminator()
{
  get_token();
  return(find_type(token)==TERM);
}

/* find the type given the word */
find_type(word)
char *word;
{
  int t;

  for(t=0; t<db_pos; t++)
    if(!strcmp(word,wdb[t].word))
      return wdb[t].type;
  return 0;
}

/* return a token from the input stream */
get_token()
{
  char *p;

  p=token;
  /* skip spaces */
  while(*t_pos==' ') t_pos++;

        if(*t_pos=='.') {
    *p++='.';
    *p='\0';
    return;
  }

  /* read word until a space or period */
  while(*t_pos!=' ' && *t_pos!='.')  {
    *p=*t_pos++;
                p++;
  }
  *p='\0';
}
```

NLPRPT.C accompanies "Natural Language Processing in C" by Herbert Schildt, BYTE, December 1987, page 269.

```
/* NLPRPT.C (4-10) */
```

```
/* Recursive descent NLP Example that reports the phrases */
/* Be sure to end all input with a period */
#include "stdio.h"

#define MAX 100

#define NOUN 1
#define VERB 2
#define ADJ 3
#define ADV 4
#define DET 5
#define PREP 6
#define TERM 7

/* structure of the word database */
struct word {
  char word[20];
  char type;
};

struct word wdb[MAX];  /* array of db structures */

int db_pos=0; /* number of entries in wdb */

char s[80];    /* holds the sentence */
char *t_pos=0;   /* points into the sentence */
char token[80];  /* contains the word */

main()
{

  setup();

  printf("Enter Sentence: ");
  gets(s);
  t_pos=s;
  if(parse()) printf("Sentence OK\n");
  else printf("Error in sentence\n");
}

setup()
{
  assert_wdb("door",NOUN);
  assert_wdb("window",NOUN);
  assert_wdb("house",NOUN);
  assert_wdb("child",NOUN);
  assert_wdb("has",VERB);
  assert_wdb("runs",VERB);
  assert_wdb("plays",VERB);
  assert_wdb("large",ADJ);
  assert_wdb("quickly",ADV);
  assert_wdb("the",DET);
  assert_wdb("a",DET);
  assert_wdb("to",PREP);
  assert_wdb(".",TERM);
}

/* place facts into database */
assert_wdb(word,type)
char *word;
int type;
{

  if(db_pos<MAX) {
    strcpy(wdb[db_pos].word,word);
    wdb[db_pos].type=type;
    db_pos++;
  }
  else printf("Word database full.\n");
}

/* Context-free recursive descent NLP parser
   that displays phrases  */
parse()
{
  char noun[80], verb[80];

  noun[0]='\0'; verb[0]='\0';
  if(!nounphrase(noun)) return 0;
  if(!verbphrase(verb)) return 0;
  if(!terminator()) return 0;
  printf("noun phrase: %s\n",noun);
```

continued

```
    printf("verb phrase: %s\n",verb);
    return 1;

  }

  /* read a noun phrase from the input stream */
  nounphrase(s)
  char *s;
  {
    char type;

    get_token();

    type=find_type(token);
    switch(type) {
      case DET:
        strcat(s,token);
        strcat(s," ");
        get_token();
        type=find_type(token);
        strcat(s,token);
        strcat(s," ");
        if(type==NOUN) return 1;
        else if(type==ADJ) {
          get_token();
          strcat(s,token);
          strcat(s," ");
          type=find_type(token);
          if(type==NOUN) return 1;
        }
        break;
      case PREP:
        strcat(s,token);
        strcat(s," ");
        return nounphrase(s);
    }
    return 0;
  }

  /* read a verb phrase */
  verbphrase(s)
  char *s;
  {
    char type,*pos, temp[80];

    get_token();

    type=find_type(token);
    if(type!=VERB) return 0;  /* must start with a verb */
    strcat(s,token);
    strcat(s," ");
    strcpy(temp,s);  /* save for backtracking */
    pos=t_pos; /* save current position for backtracking */

    /* verb + adverb + NP */
    if(verb_adv_np(s)) return 1;

    /* verb +NP */
    t_pos=pos;  /* back up */
    strcpy(s,temp);
    if(verb_np(s)) return 1;

    /* verb+adverb -- no NP */
    t_pos=pos;
    strcpy(s,temp);
    if(verb_adv(s)) return 1;

    /* just verb */
    return 1;  /* error in verb phrase */
  }

  verb_np(s)
  char *s;
  {
    /* verb + NP */
    return nounphrase(s);
  }

  verb_adv_np(s)
  char *s;
  {
    char type, temp[80];

    get_token();
    type=find_type(token);
    strcat(s,token);
```

```
    strcat(s," ");
    temp[0]='\0';

    if(type==ADV && nounphrase(temp)) (
      strcat(s,temp);
      return 1;
    )
    return 0;
)

verb_adv(s)
char *s;
(
  char type;

  get_token();
  type=find_type(token);
  strcat(s,token);
  strcat(s," ");

  return (type==ADV);
)

terminator()
(
  get_token();
  return(find_type(token)==TERM);
)

/* find the type given the word */
find_type(word)
char *word;
(
  int t;

  for(t=0; t<db_pos; t++)
    if(!strcmp(word,wdb[t].word))
      return wdb[t].type;
  return 0;
)

/* return a token from the input stream */
get_token()
(
  char *p;

  p=token;
  /* skip spaces */
  while(*t_pos==' ') t_pos++;

        if(*t_pos=='.') (
    *p++='.';
    *p='\0';
    return;
  )

  /* read word until a space or period */
  while(*t_pos!=' ' && *t_pos!='.')  (
    *p=*t_pos++;
                p++;
  )
  *p='\0';
)
```

Inside IBM

EXKEY.BAT is from "Better Batch Files Through Assembler", by William J. Claff, BYTE, IBM Special Issue, 1987, page 159.

```
EXKEY.BAT

ECHO OFF
:prompt
CLS
ECHO Press 1    for program A
ECHO Press 2    for program B
ECHO Press Esc to exit to DOS
:getkey
key
IF ERRORLEVEL 49 IF NOT ERRORLEVEL 50 GOTO a
IF ERRORLEVEL 50 IF NOT ERRORLEVEL 51 GOTO b
IF ERRORLEVEL 27 IF NOT ERRORLEVEL 28 GOTO exit
GOTO getkey
:a
ECHO Execute program A
PAUSE
GOTO prompt
:b
ECHO Execute program B
PAUSE
GOTO prompt
:exit
```

INSERT.ASM is from "Pipes and Filters", by Paul Baker, BYTE, IBM Special Issue, 1987, page 215.

```
;-----------------------------------------------------------------
;                   INSERT.ASM
;
;       IN MICROSOFT MACRO ASSEMBLER SOURCE CODE
;
; PURPOSE:
;       Insert data or blank columns into a text file.  Not only
;       string data, but the carriage return [CR], line feed [LF],
;       and/or form feed [FF] characters can be inserted.  Can be
;       used with the pipe or redirection function.
;
; SYNTAX: INSERT /n/ [<"data string">] [CR] [FF] [LF] /
;       Where n = column or character position to start the
;       insert (255 max).  CR = carriage return, LF = line feed,
;       and FF = form feed.
;
; EXAMPLE: INSERT /1/"NEW DATA"CRLF/
;       Would insert the words NEW DATA and then a carriage return
;       and line feed at the beginning of each line of the input
;       file.  The result would be that NEW DATA would be
;       inserted between each line of the input file.
;
; EXAMPLE: INSERT /9/CRLF/ < OLDDATA.TXT > NEWDATA.TXT
;       Would insert CR and LF at position 9 on each line of the
;       file OLDDATA.TXT and store it in a file named NEWDATA.TXT.
;
;   <C>  DEC. 1986      PAUL BAKER  CLEVELAND TN
;-----------------------------------------------------------------
                PAGE ,132
dosint          MACRO function          ; call the DOS interrupt
                MOV     AH,function     ; put function number in AH
                INT     21h
                ENDM
```

continued

```
code            SEGMENT byte public 'code'
                ASSUME CS:CODE,DS:DATA,SS:STACK
                ORG     0100H
STRT:
                PUSH    DS              ; DO HOUSEKEEPING TO
                SUB     AX,AX           ;ALLOW RETURN TO DOS
                PUSH    AX
                MOV     AX,DATA
                MOV     ES,AX           ; set ES to top of data seg.
                CLD
                XOR     CX,CX           ; clear CX
                MOV     BX,CX           ; clear BX
                MOV     SI,0080h        ; point to PSP
                LODSB                   ; find how many params
                MOV     CL,AL           ; CL has count of params
                CMP     CL,00h          ; if no params then show
                JNZ     not_zero        ; use instructions.
                MOV     DX,OFFSET use_msg
                PUSH    ES
                POP     DS              ; get local data segment.
                dosint  09h             ; display help screen.
                IRET                    ; return to DOS.
not_zero:       CMP     CL,07h          ; 7 is min. # of params.
                JB      bad_param       ; if < 7 then error.
                INC     SI              ; skip first space
                LODSB                   ; see if / is present
                CMP     AL,2Fh
                JNE     bad_param       ; if no / then error.
                SUB     CX,0002h        ; adjust param count.
                MOV     DI,OFFSET strt_data     ; set to top of buffer.
get_strt:       LODSB                   ; load next param.
                DEC     CX
                CMP     AL,2Fh          ; if / then must be end
                JZ      get_insert      ; of first parameter
                CMP     CX,0000h        ;
                JBE     bad_param       ; if last  param then leave.
                CMP     AL,30h          ; if not a number between
                JB      bad_param       ; 0 & 9 then exit
                CMP     AL,39h          ; and give error
                JG      bad_param       ; message.
                INC     BL              ; bump digit count.
                CMP     BL,03h          ; three digits max.
                JG      bad_param       ; if more than 3 then error.
                SUB     AL,30h          ; convert to binary.
                CALL    store_byte      ; store each digit.
                JMP     get_strt        ; get next param
get_insert:     MOV     DI,OFFSET insert_data   ; load top of buffer
                CMP     BL,00h          ; if no first param then
                JLE     bad_param       ; send error message.
                MOV     BH,00h          ; BH will count insert bytes.
                LODSB                   ; get next param.
                DEC     CX              ; reduce param count.
                CMP     AL,22h          ; in quotes ?
                JZ      in_quotes       ; if so then go process.
                CMP     AL,2Fh          ; if another / then
                JZ      bad_param       ; invalid parameter.
no_quotes:      MOV     AH,AL           ; move low byte to high byte
                LODSB                   ; load next byte.
                DEC     CX              ; reduce param count.
                CALL    caps            ; force upper case.
                CMP     AX,4352h        ; if CR.
                JZ      cr
                CMP     AX,4C46h        ; if LF.
                JZ      lf
                CMP     AX,4646h        ; if FF.
                JZ      ff
                JMP     bad_param       ; else must be error.
bad_param:      MOV     DX,OFFSET err1  ; point to error message
                MOV     CX,2Fh          ; send 47 bytes.
                MOV     BX,0002h        ; send to error output.
                PUSH    ES
                POP     DS              ; get new data segment
                dosint  40h             ; display it
leave:  IRET            ; RETURN TO DOS
cr:             MOV     AL,0Dh          ; load CR byte.
                JMP     put_insert
lf:             MOV     AL,0Ah          ; load LF byte.
                JMP     put_insert
ff:             MOV     AL,0Ch          ; load FF byte.
put_insert:     CALL    store_byte      ; store insert info.
                INC     BH              ; bump insert byte count.
                LODSB                   ; get next param.
                CMP     AL,2Fh          ; if / then must be end
                JZ      process         ; of parameters.
                CMP     AL,22h          ; if quotes then
```

```
                JZ      in_quotes       ; is in quotes again.
                JMP     no_quotes       ; else loop back.
in_quotes:      LODSB
                CMP     AL,22h          ; if " then out of quotes.
                JNE     in_quotes1      ; if not " then go on.
                LODSB                   ; else get next after ".
                CMP     AL,2Fh          ; if / then must be end
                JZ      process
                JMP     no_quotes
in_quotes1:     CMP     AL,2Fh          ; if / then must be error
                JZ      bad_param       ; since no closing ".
                CALL    store_byte      ; store current byte.
                INC     BH              ; bump insert byte count.
                JMP     in_quotes       ; loop back.
;
; ----------  PROCESS INCOMING DATA  -------------------
;
process:
                PUSH    ES              ; load local data pointer.
                POP     DS
                MOV     insert_len,BH   ; store insert length.
                MOV     digit_cnt,BL    ; store digit count.
                MOV     AX,01h          ; start AX @ 1
                XOR     CX,CX           ; clear CX register.
                MOV     CL,digit_cnt    ; load loop count.
loop1:      MUL         mult10          ; multiply by 10
                LOOP    loop1           ; create multiplier
                MOV     BX,AX           ; save multiplier.
                MOV     CL,digit_cnt    ; load loop count.
                MOV     SI,OFFSET strt_data     ; point to 1'st digit
loop2:      MOV         AX,BX           ; get multiplier.
                DIV     mult10          ;
                MOV     AH,00h          ; clear remainder.
                MOV     BX,AX           ; update multiplier.
                MUL     BYTE PTR [SI]   ; multiply current digit.
                ADD     strt_col,AL     ; update start col #.
                INC     SI              ; bump pointer.
                LOOP    loop2
get_ready:      XOR     BX,BX           ; load handle 00
                dosint  45h             ; get file duplicate.
                MOV     BP,AX           ; set base pointer to handle.
                dosint  3Eh             ; close file.
;               MOV     BX,0002h        ;
;               dosint  45h             ; file duplicate.
read_data:      CLD
                MOV     DX,OFFSET data_buf      ; store in data_buf
                MOV     CX,800h         ; set to read 800h bytes.
                MOV     BX,BP           ; set BX to file handle.
                dosint  3Fh             ; read input data.
                OR      AX,AX           ; data read ?
                JNZ     more_data
no_data:        IRET
more_data:      MOV     CX,AX           ; CX has count of bytes read.
                MOV     SI,DX           ; point to top of data.
get_byte:       MOV     BL,strt_col     ; see if this is
                CMP     BL,col_cnt      ; where the insert goes.
                JNZ     no_hit          ; if not, then no hit.
                CALL    make_insert     ; else make insert.
no_hit: LODSB                   ; get first byte.
                CMP     AL,1Ah          ; if end of file, quit.
                JZ      no_data
                CMP     AL,0Dh          ; cr ?
                JNZ     no_cr           ; if not, go on.
                MOV     col_cnt,01h     ; else reset column count.
                MOV     DL,AL           ; send the CR
                dosint  02h
                DEC     CX              ; reduce byte count.
                JMP     no_hit          ; get next byte.
no_cr:      CMP         AL,09h          ; check for tab.
                JNZ     no_tab
                CALL    tab             ; if so, call tab routin.
                JMP     get_byte        ; loop back.
no_tab:     CMP         AL,1Fh          ; do not count anything
                JBE     no_count        ; else below 20h.
send_byte:      INC     col_cnt         ; bump column count.
no_count:       MOV     DL,AL           ; send byte to disply.
                dosint  02h
                DEC     CX
go_back:        JCXZ    read_data
                JMP     get_byte
                IRET
```

continued

```
;
; ---------- MAKE INSERT  ROUTINE  ------------------------
;
make_insert:    CMP     insert_len,01h  ; if nothing to insert
                JB      exit_insert     ; then leave.
                PUSH    CX              ; save CX and
                PUSH    SI              ; and SI info.
                MOV     SI,OFFSET insert_data   ; point to data.
                XOR     CX,CX           ; load CX with
                MOV     CL,insert_len   ; loop count.
insert_loop:    LODSB                   ; load next byte.
                MOV     DL,AL
                dosint  02h             ; send byte out.
                LOOP    insert_loop
                POP     SI              ; restore SI and
                POP     CX              ; CX data.
exit_insert:    RET
;
;---------------  TAB ROUTINE  --------------------------
;
tab:            MOV     CX,0008h        ; expand tabs count.
                MOV     DL,20h          ; load space byte.
tab_loop:       CMP     BL,col_cnt      ; time to make insert ?
                JNZ     not_yet         ;
                CALL    make_insert     ; if so, then call insert.
                MOV     DL,20h          ; reload space byte.
not_yet:        dosint  02h             ; send byte.
                INC     col_cnt         ; bump column count.
                LOOP    tab_loop        ; loop back.
                RET
;
; --------------  CAPS  ROUTINE  -------------------------
;
caps:       CMP         AL,61h          ; convert to
               JB       exit_caps       ; all caps.
               CMP      AL,7Ah
               JG       exit_caps
               AND      AX,5F5Fh
exit_caps:     RET
;
; ----------  STORE BYTE ROUTINE  ----------------------------
;
store_byte:     PUSH    DS              ; store current DS
                PUSH    ES              ; local data segment info.
                POP     DS              ; exchange them.
                STOSB                   ; store local data.
                PUSH    DS
                POP     ES              ; restore original
                POP     DS              ; positions.
                RET
CODE    ENDS

;
; ------------   STACK  SEGMENT  ----------------------------
;
stack   SEGMENT STACK 'stack'
                DB      32 DUP(?)
stack   ENDS

;
; -------------  DATA SEGMENT  --------------------------------
;
data            SEGMENT byte public 'DATA'
err1            DB  'INSERT : error - missing or invalid paramaters $'
strt_data       DB      3h DUP (00h)    ; start position as entered
strt_col        DB      00              ; column to start insert
col_cnt DB      01h             ; current column count.
digit_cnt       DB      00h             ; # of digits in 1'st param
insert_len      DB      00h             ; length of insert data string
insert_data     DB      80h DUP(?)      ; buffer for insert data
data_buf        DB      800h DUP(?)     ; buffer for input data.
mult10  DB      0Ah             ; times 10 multiplier.
use_msg DB      13,10,'PURPOSE :',13,10,10
  DB    '   Insert data or blank columns into a '
  DB    'text file. Not only string data, but',13,10
  DB    '   the Carriage Return [CR], Line Feed [LF]  and/or'
  DB    ' Form Feed [FF] characters ',13,10
  DB    '   can be inserted. Can be used with the pipe or '
  DB    ' re-direction function.',13,10,10
  DB   'SYNTAX : INSERT /n/ [<"data string">] [CR] [FF] [LF] /',13,10
  DB    10,'   Where n = column or character position'
  DB    ' to start the insert. (255 max.)',13,10
  DB    '   CR = Carriage Return, LF = Line Feed and  FF = Form Feed'
  DB    13,10,10,'EXAMPLE : INSERT /1/"NEW DATA"CRLF/',13,10,10
  DB    '   Would insert the words NEW DATA and then a carriage'
  DB    ' return and line feed ',13,10
```

```
   DB    '   at the begining of each line of the input file.'
   DB    'The result would be that',13,10
   DB    '   NEW DATA would be inserted between each line'
   DB    ' of the input file.',10,13,10
   DB    'EXAMPLE :  INSERT /9/CRLF/ < OLDDATA.TXT > NEWDATA.TXT'
   DB    13,10,10,'   Would insert CR and LF at position 9 on each'
   DB    ' line of the file OLDDATA.TXT ',13,10
   DB    '   and store it in a disk file'
   DB    ' named NEWDATA.TXT.',13,10,10
   DB    '           <C>  DEC. 1986   PAUL BAKER $'
data            ENDS
     END STRT
```

KEYIN.ASM is from "Better Batch Files Through Assembler", by William J. Claff, BYTE, IBM Special Issue, 1987, page 159.

```
CODE    SEGMENT
        ASSUME  CS:CODE          ;<- ASSUMES FOR .COM FILE
        ASSUME  DS:CODE
        ASSUME  ES:CODE
        ASSUME  SS:CODE
        ORG     00100H           ;<- REQUIRED FOR .COM FILE
IP      LABEL   NEAR             ;   (USED ON END STATEMENT)
        JMP     START
CURSOR  DW      ?
START   LABEL   NEAR
        MOV     AH,3             ;<- GET CURSOR MODE
        INT     010H
        CMP     CX,00067H        ;<- CHECK FOR BUG
        JNE     NOBUG
        MOV     CX,00607H
NOBUG   LABEL   NEAR
        MOV     CURSOR,CX
        MOV     AH,1             ;<- TURN CURSOR OFF
        MOV     CX,02000H
        INT     010H
FLUSH   LABEL   NEAR             ;<- FLUSH THE KEYBOARD BUFFER
        MOV     AH,1
        INT     016H
        JZ      FLUSHED
        MOV     AH,0
        INT     016H
        JMP     FLUSH
FLUSHED LABEL   NEAR
        MOV     AH,0             ;<- WAIT FOR A KEYSTROKE
        INT     016H
        OR      AL,AL
        JNZ     REGULAR
        MOV     AL,AH            ;<- FUNCTION AND OTHER SPECIAL KEYS
        OR      AL,10000000B     ;   TURN ON HIGH-BIT
        JMP     SHORT DONE
REGULAR LABEL   NEAR             ;<- REGULAR KEY
        CMP     AL,'a'           ;   CONVERT TO UPPER-CASE
        JB      DONE
        CMP     AL,'z'
        JA      DONE
        ADD     AL,'A'-'a'
DONE    LABEL   NEAR
        PUSH    AX
        MOV     AH,1
        MOV     CX,CURSOR
        INT     010H
        POP     AX
EXIT    LABEL   NEAR             ;<- EXIT
        MOV     AH,04CH
        INT     021H
CODE    ENDS
        END     IP               ;<- REQUIRED FOR .COM FILE
```

continued

MAKECOM.BAT is from "Better Batch Files Through Assembler", by William J. Claff, BYTE, IBM Special Issue, 1987, page 159.

```
ECHO OFF
IF "\"%1 == "\" GOTO error0
IF NOT EXIST %1.asm GOTO error1
IF EXIST %1.obj erase %1.obj
masm %1;
IF ERRORLEVEL 1 GOTO error2
IF EXIST %1.exe erase %1.exe
link %1;
IF ERRORLEVEL 1 GOTO error3
erase %1.obj
IF EXIST %1.com erase %1.com
exe2bin %1 %1.com
IF NOT EXIST %1.com GOTO error4
erase %1.exe
ECHO %1 MASMed, LINKed and EXE2BINed successfully.
GOTO exit
:error0
ECHO Usage: makecom source_asm
GOTO exit
:error1
ECHO %1.asm does not exist.
goto EXIT
:error2
ECHO %1 did not MASM successfully.
goto EXIT
:error3
ECHO %1 did not LINK successfully.
goto EXIT
:error4
ECHO %1 did not EXE2BIN successfully.
:exit
```

MSDRIVER.ASM is from "Application Input Drivers", by Jeremy Sagan, BYTE, IBM Special Issue, 1987, page 143.

```
;
;
;
; ====================== CODE SEGMENT DEFN ======================
;
CSEG    segment public para 'code'
;
; Set to MICROSOFT = 0 to make mousesys  sample program
; Set to MICROSOFT = 1 to make microsoft sample program
;
MICROSOFT       = 0
;
        ORG     0
;
ZERO    LABEL   WORD
;
        ORG     100h            ; For sample program only
;
;
        ASSUME CS:CSEG, DS:CSEG, ES:NOTHING, SS:NOTHING
;
START:
        JMP     SAMPLECODE
;
; These are the available functions:
;
;        function 0 = initialize mouse
;        function 1 = return button status
;        function 2 = return relative motion
;        function 3 = de-initialize mouse
;        function 4 = return current serial port
;
        IF MICROSOFT
        INCLUDE MICROSOF.ASM
        ELSE
        INCLUDE MOUSESYS.ASM
        ENDIF
```

```
;
VIDEO   =       10H
;
MAXX    DW      80
MAXY    DW      22
MYX     DW      0
MYY     DW      0
MBSTAT  DB      0FFH
CVM     DB      0
;
BUT1MES DB      'Button one:  $'
BUT2MES DB      'Button two:  $'
BUT3MES DB      'Button three:$'
PRESMES DB      'pressed $'
RELMES  DB      'released$'
ERRMES  DB      'Mouse or Mouse software not installed$'
;
ERR_NODEVICE:
        MOV     DX,OFFSET ERRMES
        CALL    PRINT
        JMP     EXIT_ROUTINE
;
SAMPLECODE:
        PUSH    CS
        POP     DS
        MOV     AH,0FH
        INT     VIDEO                   ; Get current video mode
        MOV     CVM,AL                  ; Save current video mode
        MOV     BYTE PTR MAXX,AH        ; store maximum cursor position
        SUB     AH,AH
        INT     VIDEO                   ; Clear screen
        MOV     DH,BYTE PTR MAXY        ; Row 22, column 0
        SUB     DL,DL
        PUSH    DX
        CALL    SET_CURSOR
        MOV     DX,OFFSET BUT1MES
        CALL    PRINT
        POP     DX
        INC     DH
        PUSH    DX
        CALL    SET_CURSOR
        MOV     DX,OFFSET BUT2MES
        CALL    PRINT
        POP     DX
        INC     DH
        CALL    SET_CURSOR
        MOV     DX,OFFSET BUT3MES
        CALL    PRINT
;
        MOV     BX,4
        CALL    ENTRY           ; Serial port in Ax
        SUB     BX,BX
        CALL    ENTRY           ; Initialize mouse
        OR      AL,AL           ; If al = zero then an error occured
        JZ      ERR_NODEVICE
SCLOOP:
        MOV     BX,2
        CALL    ENTRY
        MOV     CX,5            ; scale x coordinate
        CWD
        IDIV    CX
        ADD     AX,MYX          ; My x cursor position
        JNS     SCLOOP1
        MOV     AX,MAXX         ; wrap to right
        DEC     AX
SCLOOP1:
        CMP     AX,MAXX
        JNG     SCLOOP2
        SUB     AX,AX           ; Wrap to left
SCLOOP2:
        MOV     MYX,AX          ; Save new x
        MOV     AX,BX           ; get delta y
        CWD
        IDIV    CX
        ADD     AX,MYY          ; My y cursor position
        JNS     SCLOOP25
        MOV     AX,MAXY         ; Wrap to bottom of screen
        DEC     AX
SCLOOP25:
        CMP     AX,MAXY
        JNG     SCLOOP3
        SUB     AX,AX           ; Wrap to top
```

continued

```
SCLOOP3:
        MOV     MYY,AX          ; Save new y
        MOV     BX,1
        CALL    ENTRY
        CMP     AL,MBSTAT
        JE      SCLOOP4
        MOV     MBSTAT,AL
        SHR     AX,1            ; put button 1 status in carry
        MOV     DH,BYTE PTR MAXY
        MOV     DL,15
        CALL    BUTTONPRINT
        SHR     AX,1
        INC     DH
        CALL    BUTTONPRINT
        SHR     AX,1
        INC     DH
        CALL    BUTTONPRINT
SCLOOP4:
        MOV     DL,BYTE PTR MYX
        MOV     DH,BYTE PTR MYY
        CALL    SET_CURSOR
        MOV     AH,1
        INT     16H             ; Keyboard input ?
        JZ      SCLOOP          ; No
        SUB     AH,AH
        INT     16H             ; Read key
        CMP     AL,27           ; <Esc> ?
        JZ      SCLOOP5
        MOV     AH,14
        INT     VIDEO
        MOV     AH,3
        SUB     BH,BH
        INT     VIDEO           ; Read cursor position
        MOV     BYTE PTR MYX,DL
        MOV     BYTE PTR MYY,DH
        JMP     SCLOOP
;
SCLOOP5:
        MOV     BX,3
        CALL    ENTRY           ; Clean up after mouse
;
EXIT_ROUTINE:
        MOV     AL,CVM
        SUB     AH,AH
        INT     VIDEO           ; Clear screen
        MOV     AX,04C00H
        INT     MSDOS           ; bye bye
;
SET_CURSOR:
        SUB     BH,BH
        MOV     AH,02
        INT     VIDEO           ; Set cursor position
        RET
;
BUTTONPRINT:
        PUSH    AX
        PUSH    DX
        PUSHF
        CALL    SET_CURSOR
        MOV     DX,OFFSET RELMES
        POPF
        JNC     BP2                     ; Carry means button pressed
        MOV     DX,OFFSET PRESMES
BP2:
        CALL    PRINT
        POP     DX
        POP     AX
        RET
;
PRINT:
        PUSH    SI
        PUSH    BX
        MOV     SI,DX
        MOV     BX,3
PRIN1:
        LODSB
        CMP     AL,'$'
        JE      PRIN2
        MOV     AH,14
        INT     VIDEO           ; Teletype routine
        JMP     PRIN1
;
PRIN2:
        POP     BX
        POP     SI
        RET
```

```
;
CSEG    ENDS
;
        END     START
```

MICROSOFT.ASM is from "Application Input Drivers", by Jeremy Sagan, BYTE, IBM Special Issue, 1987, page 143.

```
;
;
MSDOS           EQU     21H     ; Ms dos interrupt call
MSYSCALL        EQU     51      ; Microsoft system call
;
;
; This is the main entry point
; all driver routines take the function call number in BX
;
;       function 0 = initialize mouse
;       function 1 = return button status
;       function 2 = return relative motion
;       function 3 = de-initialize mouse
;       function 4 = return current serial port
;
; Normally this would be a far procedure but to avoid getting into
; all the intricasies of loading and calling drivers I've converted
; ENTRY to a near procedure and combined it with the sample program.
;
ENTRY   PROC    NEAR
        CLD                         ; go in the forward direction
        PUSH    DS                  ; save callers segment
        PUSH    CS                  ; make this segment addressable
        POP     DS
        SHL     BX,1                ; point to routine
        CALL    ROUTINES[BX]        ; and call it through table
        POP     DS                  ; restore users segment
        RET                         ; return far to caller
ENTRY   ENDP
;
        DB      'Microsoft',00  ; name
MOUSEF  DB      0
;
ROUTINES LABEL WORD
        DW      ISERIAL             ; function 0 = initialize mouse
        DW      BUTTONSTAT          ; function 1 = return button status
        DW      MOTIONCOUNT         ; function 2 = return relative motion
        DW      DSERIAL             ; function 3 = de-initialize mouse
        DW      GSERIAL             ; function 4 = return current serial port
        DW      RETADR              ; function 5 = reserved
        DW      RETADR              ; function 6 = reserved
        DW      RETADR              ; function 7 = reserved
;
;
BUTTONSTAT:
        SUB     AX,AX
        TEST    MOUSEF,0FFH     ; if no mouse then return zero
        JZ      BSTAT2
        MOV     AX,5
        SUB     BX,BX
        INT     MSYSCALL            ; Read button press info
        SUB     AH,AH               ; make ax contain press info
BSTAT2:
;
; for Microsoft mouse the exit routine does nothing
;
DSERIAL:
RETADR:
        RET
;
; This routine returns delta mouse movement
;       Ax = delta x
;       Bx = delta y
;
MOTIONCOUNT:
        SUB     AX,AX
        MOV     BX,AX
        TEST    MOUSEF,0FFH
        JZ      MCOUNT2
```

continued

```
        PUSH    DX
        PUSH    CX
        MOV     AX,11
        INT     MSYSCALL
        MOV     AX,CX           ; place into appropriate
        MOV     BX,DX           ; registers
        POP     CX
        POP     DX
MCOUNT2:
        RET
;
GSERIAL:
        SUB     AX,AX           ; No serial port needed
        RET                     ; for Microsoft Mouse
;
ISERIAL PROC    NEAR
        PUSH    ES
        PUSH    BX
        MOV     AX,03500h+MSYSCALL      ; 35h = get INTerrupt vector
        INT     MSDOS                   ; returns in ES:BX
        MOV     AX,ES
        OR      AX,BX                   ; Does interupt point at 0:0?
        MOV     AX,0                    ; if it does then
        POP     BX                      ; Mouse software is not loaded
        POP     ES                      ; and therefore cannot be called
        JZ      SER2                    ; to do initialization.
        INT     MSYSCALL
SER2:
        MOV     MOUSEF,AL
        RET                             ; returns al non 0 if mouse exists.
ISERIAL ENDP
;
```

MSYSMOUSE.ASM is from "Application Input Drivers", by Jeremy Sagan, BYTE, IBM Special Issue, 1987, page 143.

```
;
        ASSUME CS:CSEG, DS:CSEG, ES:NOTHING, SS:NOTHING
;
SERIAL  EQU     14H
MSDOS   EQU     21H
;
;
; This is the main entry point
; all driver routines take the function call number in BX
;
;        function 0 = initialize mouse
;        function 1 = return button status
;        function 2 = return relative motion
;        function 3 = de-initialize mouse
;        function 4 = return current serial port
;
; Normally this would be a far procedure but to avoid getting into
; all the intricasies of loading and calling drivers I've converted
; ENTRY to a near procedure and combined it with the sample program.
;
ENTRY   PROC    NEAR
        CLD                     ; go in the forward direction
        PUSH    DS              ; save callers segment
        PUSH    CS              ; make this segment addressable
        POP     DS
        SHL     BX,1            ; point to routine
        CALL    ROUTINES[BX]    ; and call it through table
        POP     DS              ; restore users segment
        RET                     ; return far to caller
ENTRY   ENDP
;
        DB      'Mouse systems',00      ; name
;
ROUTINES LABEL WORD
        DW      ISERIAL         ; function 0 = initialize mouse
        DW      BUTTONSTAT      ; function 1 = return button status
        DW      MOTIONCOUNT     ; function 2 = return relative motion
        DW      DSERIAL         ; function 3 = de-initialize mouse
        DW      GSERIAL         ; function 4 = return current serial port
        DW      RETADR          ; function 5 = reserved
        DW      RETADR          ; function 6 = reserved
        DW      RETADR          ; function 7 = reserved
```

```
;
COMNUM  DW      00              ; com#
;
NEWX    DW      00              ; New x coordinate
NEWY    DW      00              ; New y coordinate
XACCUM  DW      0               ; Old x coordinate
YACCUM  DW      0               ; Old y coordinate
BSTAT   DB      07H             ; button status byte
CPORT   DW      03F8H           ; communications port address
PCOUNT  DB      0               ; packet counter
IMSK    DB      0EFH            ; interrupt mask
;
; This is the heart of the code.
; The serial interrupt handler. This code catches serial bytes and maintains
; a running total of delta x and delta y values.
;
;
ISR:
        STI                     ; Ints back on
        PUSH    AX              ; Save all registers used
        PUSH    BX
        PUSH    DX
        PUSH    DS
        PUSH    CS
        POP     DS              ; make Code SEGment addressable
        MOV     DX,CPORT        ; get port address
        ADD     DX,5            ; Status
        IN      AL,DX           ; Read status
        MOV     AH,AL           ; Save in Ah
        SUB     DX,5            ; back to data port address
        IN      AL,DX           ; get byte from port
        AND     AH,01EH         ; mask error bits of status
        JNZ     ISR3            ; jmp if error
;
; Jump if an error has occured on the serial line! most likely an overrun
; error caused by interrupts cleared for long periods of time.
; This will be handled simply by clearing the packet counter.
;
ISR2:
        CMP     PCOUNT,0        ; Is this the first byte of packet ?
        JNE     ISR25           ; no so accumulate.
        MOV     AH,AL           ; it is the first byte so check certain
        AND     AH,0F8H         ; bits to se if we're in sync with the
        CMP     AH,080H         ; data stream. If we're not then
        JNZ     ISR4            ; we'll just return
        MOV     BSTAT,AL        ; we are in sync so stuff button status byte
ISR25:
        MOV     BL,PCOUNT       ; get packet counter
        INC     PCOUNT          ; increment for next serial interrupt
        OR      BL,BL           ; if it's zero we're done
        JZ      ISR4
;
        CBW                     ; Convert delta byte to delta word
        TEST    BL,1            ; Check if odd or even. odd = x values
        JZ      ADDY            ; even = y values
        ADD     XACCUM,AX       ; add to running x accumulator
        JMP     SHORT ISR29
ADDY:
        ADD     YACCUM,AX       ; or add to running y accumulator
ISR29:
        CMP     BL,4            ; end of packet
        JB      ISR4            ; no
ISR3:
        MOV     PCOUNT,0        ; yes, so reset packet counter
ISR4:
        CLI
        MOV     AL,020H         ; must issue EOI
        OUT     020H,AL
        POP     DS
        POP     DX
        POP     BX
        POP     AX              ; Restore registers
        IRET                    ; and return from interrupt
;
; This table maps button values into ones appropriate for returning
; to application
;
BUTMAP  DB      07,03,05,01,06,02,04,00
;
; This routine returns button status = ax
; a 1 bit indicates button presses
;
```

continued

```
BUTTONSTAT:
        PUSH    BX
        MOV     AL,BSTAT
        AND     AX,7
        MOV     BX,OFFSET BUTMAP  ; convert to Microsoft format
        XLAT
        POP     BX
RETADR:
        RET
;
; Motion Count routine
;       on entry:
;  Ax=cursor x,Bx=cursor y (Ignored by this driver)
;       on exit:
;  Ax=delta  x,Bx=delta y
;
MOTIONCOUNT:
        CALL    QREADPACKET     ; Read a packet
        MOV     BX,NEWY         ; return y coordinates
        NEG     BX              ; positive coordinate move down the screen
        MOV     AX,NEWX         ; return x
        RET
;
; clean up the serial port interrupts and masks
;
DSERIAL:
        CLI
        IN      AL,021H         ; Read interupt mask
        MOV     AH,IMSK         ; clear appropriate int
        NOT     AH              ; by setting bits
        OR      AL,AH
        OUT     021H,AL         ; write it out
        MOV     DX,CPORT        ; get port address
        ADD     DX,3            ; line control register
        IN      AL,DX           ; fetch it
        AND     AL,07FH         ; set low to access interrupt
        OUT     DX,AL           ; enable register
        SUB     DX,2            ; point at interrupt enable register
        SUB     AL,AL           ; clear
        OUT     DX,AL           ; it
        ADD     DX,3            ; and clear
        OUT     DX,AL           ; modem control register
        STI                     ; finished
        RET
;
GSERIAL:
        MOV     AX,COMNUM       ; returns com#
        INC     AX
        RET
;
; This code intializes the mouse systems serial mouse
; it takes the com number in Ax (1 = com1 2 = com2)
;
ISERIAL PROC    NEAR
        PUSH    CX
        DEC     AX
        MOV     COMNUM,AX               ; Save com#
        MOV     DX,AX
        MOV     AX,087H                 ; 12K BAUD
        INT     SERIAL                  ; let bios initialize baud rate and stuf
;
        PUSH    BX
        PUSH    DX
        PUSH    CS
        MOV     AX,040H                 ; point at bios data segment
        MOV     DS,AX
        MOV     BX,DX
        SHL     BX,1
        MOV     DX,ZERO[BX]             ; Get port address at 40:0 or 40:2
        POP     DS
        MOV     CPORT,DX                ; save it
        CLI
        MOV     DX,OFFSET ISR           ; stick the serial interrupt handler
        MOV     AL,00CH                 ; in either Int 0Ch, or 0Bh
        MOV     AH,BYTE PTR COMNUM
        AND     AH,1                    ; only 2 Interrupts available
        SUB     AL,AH
        MOV     AH,025H                 ; set interrupt request
        INT     MSDOS
;
        IN      AL,021H                 ; mask the interrup controller
        MOV     AH,0EFH
        CMP     COMNUM,0
        JZ      ISERIAL2
        MOV     AH,0F7H
```

```
ISERIAL2:
        MOV     IMSK,AH                 ; Save for later
        AND     AL,AH
        OUT     021H,AL
;
        MOV     DX,CPORT                ; get port address
        ADD     DX,3                    ; Line control register
        IN      AL,DX
        AND     AL,07FH                 ; make interrupt enable register
        OUT     DX,AL                   ; Addressable
        JMP     $+2
        SUB     DX,2                    ; Point at it
        MOV     AL,1                    ; set the data available int
        OUT     DX,AL
        ADD     DX,3                    ; Modem control register
        MOV     AL,08                   ; set it
        OUT     DX,AL
        STI
        MOV     DX,CPORT
        IN      AL,DX                   ; Clear receive buffer
        POP     DX
        POP     BX
        POP     CX
        MOV     AL,-1                   ; mouse available.
        RET
ISERIAL ENDP
;
;
QREADPACKET:
QR1:
        CLI
;
        NOP
        MOV     AX,XACCUM       ; get x accumulator
        MOV     NEWX,AX         ; move to new delta x
        MOV     AX,YACCUM       ; get y
        MOV     NEWY,AX         ; to new y
        SUB     BX,BX
        MOV     XACCUM,BX       ; clear
        MOV     YACCUM,BX       ; accumulators
;
        STI
QREXIT:
        RET
;
```

ONEKEY.ASM is from "Better Batch Files Through Assembler", by William J. Claff, BYTE, IBM Special Issue, 1987, page 159.

```
;=======DUMMY SEGMENTS
ENV     SEGMENT AT 0FFFFH       ;<- THE ENVIRONMENT
STRINGS DB      ?
ENV     ENDS
MEM     SEGMENT AT 0FFFFH       ;<- A MEMORY BLOCK
MEMTYPE DB      ?
MEMID   DW      ?
MEMSIZE DW      ?
MEM     ENDS
PSP     SEGMENT AT 0FFFFH       ;<- COMMAND.COM
        ORG     0002CH          ;<- ENVIRONMENT SEGMENT
ENVSEG  DW      ?
PSP     ENDS
;=======CODE SEGMENT
CODE    SEGMENT
        ASSUME  CS:CODE         ;<- ASSUMES FOR .COM FILE
        ASSUME  DS:CODE
        ASSUME  ES:CODE
        ASSUME  SS:CODE
        ORG     00016H          ;<- COMMAND.COM SEGMENT
CMDSEG  DW      ?
        ORG     00100H          ;<- REQUIRED FOR .COM FILE
IP      LABEL   NEAR            ;   (USED ON END STATEMENT)
        JMP     START
CURSOR  DW      ?
START   LABEL   NEAR
;-------TURN CURSOR OFF
        MOV     AH,3            ;<- GET CURSOR MODE
```

continued

```
        INT     010H
        CMP     CX,00067H       ;<- CHECK FOR BUG
        JNE     NOBUG
        MOV     CX,00607H
NOBUG   LABEL   NEAR
        MOV     CURSOR,CX
        MOV     AH,1            ;<- TURN CURSOR OFF
        MOV     CX,02000H
        INT     010H
;-------LOCATE ENVIRONMENT
        MOV     DX,CMDSEG       ;<- LET DS=COMMAND.COM SEGMENT
        MOV     DS,DX
        ASSUME  DS:PSP
        MOV     AX,ENVSEG       ;<- CHECK ENVIRONMENT SEGMENT
        OR      AX,AX
        JNZ     HAVEIT
        MOV     AX,DS           ;<- ROOT COMMAND.COM
        DEC     AX              ;   WALK MEMORY BLOCKS FORWARD
        MOV     DS,AX           ;   TO THE ENVIRONMENT
        ASSUME  DS:MEM
NEXTSEG LABEL   NEAR
        CMP     MEMTYPE,'M'
        JNE     ERROR1          ;<- !ERROR! SHOULD NOT OCCUR
        MOV     AX,DS
        INC     AX
        ADD     AX,MEMSIZE
        MOV     DS,AX
        ASSUME  DS:MEM
        CMP     MEMID,DX
        JNE     NEXTSEG
        MOV     AX,DS
        INC     AX
HAVEIT  LABEL   NEAR
;-------LOCATE K= IN THE ENVIRONMENT
        MOV     ES,AX
        ASSUME  ES:ENV
        XOR     AL,AL
        MOV     CX,08000H
        XOR     DI,DI
SEARCH  LABEL   NEAR
        REPNE   SCAS    STRINGS
        CMP     STRINGS[DI],AL
        JE      ERROR2          ;<- !ERROR! K= MISSING
        CMP     STRINGS[DI],'K'
        JNE     SEARCH
        CMP     STRINGS[DI][1],'='
        JNE     SEARCH
        ADD     DI,2            ;<- POSITION AFTER = SIGN
;-------GET KEYSTROKE. RESTRICT TO ASCII ! THROUGH ~
FLUSH   LABEL   NEAR            ;<- FLUSH THE KEYBOARD BUFFER
        MOV     AH,1
        INT     016H
        JZ      FLUSHED
        MOV     AH,0
        INT     016H
        JMP     FLUSH
FLUSHED LABEL   NEAR
GETKEY  LABEL   NEAR
        MOV     AH,0            ;<- WAIT FOR A KEYSTROKE
        INT     016H
        OR      AL,AL
        JZ      GETKEY
        CMP     AL,'!'
        JB      GETKEY
        CMP     AL,'~'
        JA      GETKEY
REGULAR LABEL   NEAR            ;<- REGULAR KEY
        CMP     AL,'a'          ;   CONVERT TO UPPER-CASE
        JB      STORE
        CMP     AL,'z'
        JA      STORE
        ADD     AL,'A'-'a'
STORE   LABEL   NEAR
        MOV     STRINGS[DI],AL  ;<- STORE IN THE ENVIRONMENT
        MOV     AH,1            ;<- RESTORE CURSOR
        MOV     CX,CURSOR
        INT     010H
;-------
        MOV     AL,0
EXIT    LABEL   NEAR            ;<- EXIT
        MOV     AH,04CH
        INT     021H
ERROR1  LABEL   NEAR            ;<- COULD NOT LOCATE ENVIRONMENT
        MOV     AL,1
        JMP     EXIT
```

```
ERROR2  LABEL   NEAR            ;<- K= MISSING FROM ENVIRONMENT
        MOV     AL,2
        JMP     EXIT
CODE    ENDS
        END     IP              ;<- REQUIRED FOR .COM FILE
```

PAL6821.LST is from "Three Bus Interface Designs for the PC", by James R. Drummond, BYTE, IBM Special Issue, 1987, page 225.

```
PAL16R4
PAL FOR 6821 TO PC BUS INTERFACE  J.R. DRUMMOND 24/1/87
CONTROLS BUS ACCESS

CLK /BS   /IOR   /IOW  RESET  PIN6  PIN7  PIN8  PIN9  GND
/OC RESB  SELB   E     C      PIN16 CS    RWB   READY VCC


IF (VCC)     /SELB  =  BS * IOR  +  BS * IOW

IF (VCC)     /RESB  = RESET

IF (/SELB)   /READY =  /CS  +  CS * /C  +  CS * C * E * IOW

IF (VCC)     /RWB   = IOW

             /CS    :=  /CS * /C  +  /CS * E
                        +  C * /E * /BS  +  /CS * C * /E * /IOR * /IOW
                        +  CS * C * /E * /IOW

             /C     :=  /C * /E  +  /CS * C * /E
                        +  CS * C * /E * /BS  +  CS * C * /E * /IOW

             /E     :=  C * /E  +  /CS * C * E
                         +  CS * C * E * /BS  +  CS * C * E * /IOR

DESCRIPTION

                        ------u------
          CLK       1-|              |-20     VCC
                      |              |
          /BS       2-|              |-19     READY
                      |              |
          /IOR      3-|              |-18     RWB
                      |              |
          /IOW      4-|            R |-17     CS
                      |              |
          RESET     5-|            R |-16     PIN16
                      |    16R4      |
          PIN6      6-|            R |-15     C
                      |              |
          PIN7      7-|            R |-14     E
                      |              |
          PIN8      8-|              |-13     SELB
                      |              |
          PIN9      9-|              |-12     RESB
                      |              |
          GND      10-|______________|-11     /OC


           CS  C  E
-->
           0   1  1
           0   1  0
            +--------------------------+-------------------------+
            |                          |                         |
      /BS + /IOR./IOW                  BS.IOW                    BS.IOR

                                       1   0   0                 1  0  0
                                       1   0   1                 1  0  1
                                       1   1   1                       <-+
                                                                 1  1  1  |BS.IOR
                                                                  -------+
                                                  <----------------
                                                  <-+
                                       1   1   0    |BS.IOW
                                                   --------+
                                <--------------------
```

continued

```
          0   0   0
          0   0   1
  <------------
```

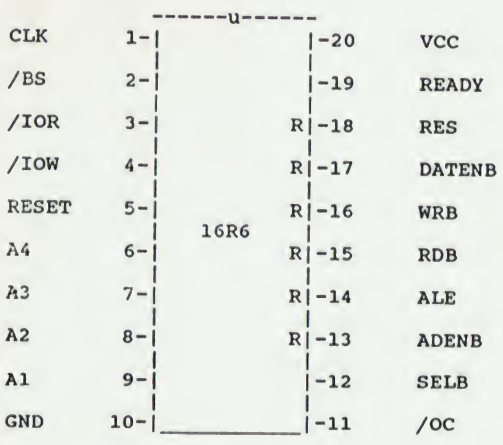
NO ACCESS WRITE CYCLE READ CYCLE

READY ASSERTED AT LOOP POINT IN BOTH CYCLES AND BEYOND

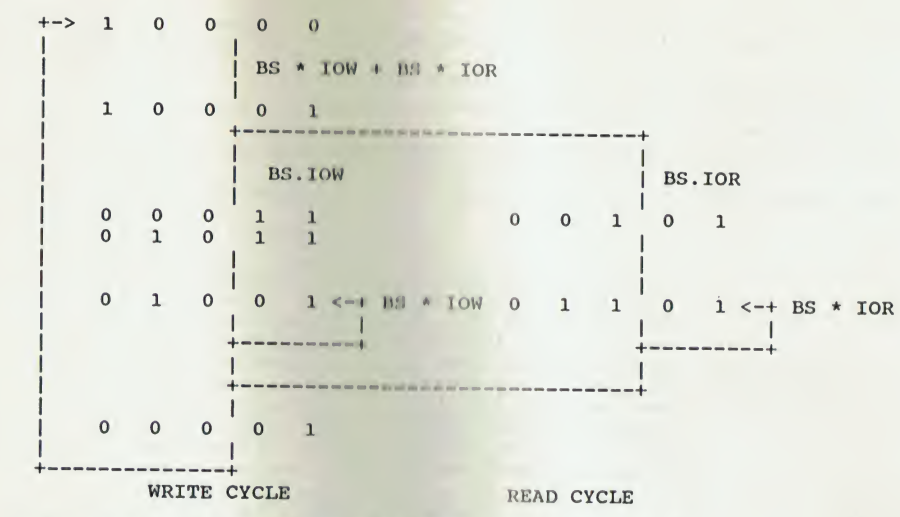
PAL810.LST is from "Three Bus Interface Designs for the PC", by James R. Drummond, BYTE, IBM Special Issue, 1987, page 225.

```
PAL16R6
PAL FOR NSC810 TO PC BUS INTERFACE  J.R. DRUMMOND 4/2/87
CONTROLS BUS ACCESS

CLK /BS   /IOR   /IOW  RESET  A4    A3     A2    A1    GND
/OC SELB  ADENB  ALE   RDB    WRB   DATENB RES   READY VCC


IF (VCC)     /SELB  =    BS * /A4 * IOR  +  BS * /A3 * IOR
                       + BS * A4 * A3 * /A2 * /A1 * IOR
                       + BS * /A4 * IOW  +  BS * /A3 * IOW
                       + BS * A4 * A3 * /A2 * /A1 * IOW

IF (/SELB)   /READY =  RDB * /WRB * /ALE  +  DATENB * /RDB * WRB * /ALE
                       + DATENB * RDB * WRB * /SELB

             /RES    :=   /RESET

             /ADENB  :=           DATENB * RDB * WRB * ALE
                        + ADENB * DATENB * RDB * WRB * /ALE


             /ALE    :=  /DATENB  +  /WRB  +  /RDB
                         +  /ADENB * DATENB * RDB * WRB * /SELB

             /RDB    :=  /ADENB * DATENB * RDB * WRB * /ALE * /SELB * IOR
                       +  /RDB * WRB * /ALE * /SELB * IOR

             /WRB    :=   /ADENB * DATENB * RDB *  WRB * /ALE * /SELB * IOW
                       +   ADENB * DATENB * RDB * /WRB * /ALE * /SELB * IOW

             /DATENB :=  RDB * /WRB * /ALE
                       +  /DATENB * RDB * WRB * /ALE * /SELB * IOW
                       +  /RDB * WRB * /ALE * /SELB * IOR
```

DESCRIPTION



ADEN DATEN RD WR /ALE



READY ASSERTED AT LOOP POINT IN BOTH CYCLES AND BEYOND

ADDRESSES 00000 -> 11001 ARE DECODED

PMODE.ASM is from "286/386 Protected-Mode Programming", by Joel Barnum, BYTE Special Issue, 1987, page 125.

```
PAGE    ,132
TITLE   PMODE

COMMENT @

     These procedures allow a protected mode program to avoid certain
     protection exceptions.

     NOTES:  1. These procedures are intended to run in protected mode
                on a 286 or 386-based computer.  Any attempt to execute
                them in real mode, or on an 8088-based PC will most likely
                result in a system crash!

             2. This file was assembled using Microsoft's MASM Version 4.0

        @

.286p                                       ;enable protected mode instructions

CODE    SEGMENT PUBLIC 'CODE'
        ASSUME CS:CODE

;--------------------------------------------------------------------------
;       check_segment_limit: Determine if an offset is usable within a segment
;               INPUT:  BX - selector for the segment
;                       CX - offset to check
;               OUTPUT: CF:  0 - selector:offset is OK
;                            1 - exception will occur if selector:offset used
;--------------------------------------------------------------------------
check_segment_limit     PROC
                        PUBLIC check_segment_limit

        PUSH    DX
        LSL     DX,BX                       ;obtain the segment's limit
        JNZ     exception_return            ;exit if bad selector
        CMP     CX,DX                       ;compare offset and limit
        JA      exception_return            ;jump if offset above limit
        CLC                                 ;selector:offset OK
        JMP     SHORT csl_exit              ;return to caller
exception_return:
        STC                                 ;bad selector or offset
csl_exit:
        POP     DX
        RET

check_segment_limit     ENDP
```

continued

```
;-----------------------------------------------------------------------------
;       check_sensitive:  Determine if we can execute sensitive instructions
;               INPUT:  None
;               OUTPUT: CF:  0 - sensitive instructions OK at current privilege
;                            1 - exception will occur if we attempt a sensitive
;                                instruction
;-----------------------------------------------------------------------------
check_sensitive         PROC
                        PUBLIC check_sensitive

        PUSH    AX
        PUSH    BX

        PUSHF                                   ;save flags (IOPL) on stack
        POP     AX                              ;copy flags to AX
        AND     AX,3000H                        ;mask all but IOPL
        SHR     AX,12                           ;right-justify IOPL
        MOV     BX,CS                           ;CPL resides in CS
        AND     BX,3                            ;mask all but CPL
        CMP     BX,AX                           ;compare CPL and IOPL
        JA      no_sensitive                    ;jump if CPL > IOPL
        CLC                                     ;sensitive instructions OK
        JMP     SHORT cs_exit
no_sensitive:
        STC                                     ;exception occurs on sensitive
cs_exit:
        POP     BX
        POP     AX
        RET

check_sensitive         ENDP

;-----------------------------------------------------------------------------
;       check_privileged: Determine if we can execute privileged instructions
;               INPUT:  None
;               OUTPUT: CF:  0 - privileged instructions OK at current privilege
;                            1 - exception will occur if we attempt a privileged
;                                instruction
;-----------------------------------------------------------------------------
check_privileged        PROC
                        PUBLIC check_privileged

        PUSH    AX
        MOV     AX,CS                           ;CPL resides in CS
        AND     AX,3                            ;mask all but CPL
        JNZ     no_privileged                   ;jump if CPL <> 0
        CLC                                     ;privileged instructions OK
        JMP     SHORT cp_exit
no_privileged:
        STC                                     ;exception occurs on privileged
cp_exit:
        POP     AX
        RET

check_privileged        ENDP

CODE    ENDS
        END
```

SPY.C is from "Spying on Windows", by Mike Geary, BYTE, IBM Special Issue, 1987, page 97.

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *\
 *  Spy.c                                                                *
 *  Windows Spy Program                                                  *
 *  Public Domain                                                        *
 *  Written by Michael Geary                                             *
 *                                                                       *
 *  This program "spies" on all the windows that are currently open in your  *
 *  Windows session, and displays a window containing all the information it *
 *  can find out about those windows.  You can scroll through this window    *
 *  using either the mouse or keyboard to view the information about the     *
 *  various windows.  The "New Spy Mission" menu item re-captures the latest *
 *  information.  This menu item is on the System menu so you can trigger it *
 *  even if the Spy window is iconic.                                    *
\* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

#define LINT_ARGS
```

```
#include <windows.h>
#include <stdio.h>
#include "spy.h"

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  The display for a single window looks like this in collapsed mode:
 *
 *  (Child|Popup|TopLevel) Window HHHH (class) (L,T;R,B) "title"
 *
 *  or like this in expanded mode:
 *
 *      (Child|Popup|TopLevel) window handle: HHHH
 *        Class name: (class name)
 *        Window title: (title text)
 *        Parent window handle: HHHH
 *        Class function, window function: HHHH:HHHH, HHHH:HHHH
 *        Class module handle, Window instance handle: HHHH, HHHH
 *        Class extra alloc, Window extra alloc: DDDDD, DDDDD
 *        Class style, Window style: HHHH, HHHHHHHH
 *        Menu handle: HHHH   -or-  Control ID: DDDDD
 *        Brush, Cursor, Icon handles: HHHH, HHHH, HHHH
 *        Window rectangle: Left=DDDDD, Top=DDDDD, Right=DDDDD, Bottom=DDDDD
 *        Client rectangle: Left=DDDDD, Top=DDDDD, Right=DDDDD, Bottom=DDDDD
 *      (blank line)
 *
 *  Total number of lines for one window display: 13
 */

#define LINES_PER_WINDOW    13
#define WINDOW_WIDTH        120

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  The INFO structure contains all the information we gather up about each
 *  window we are spying on.  We allocate an array of INFO structures in the
 *  global heap, with one entry for each window in the system.
 */

#define CLASSMAX    30
#define TITLEMAX    50

typedef struct {
    HWND    winHWnd;                /* Window handle */
    char    winClass[CLASSMAX];     /* Class name */
    HBRUSH  winBkgdBrush;           /* Background brush handle */
    HCURSOR winCursor;              /* Cursor handle */
    HICON   winIcon;                /* Icon handle */
    HANDLE  winClassModule;         /* Module handle for owner of class */
    WORD    winWndExtra;            /* Extra data allocated for each window */
    WORD    winClsExtra;            /* Extra data allocated in class itself */
    WORD    winClassStyle;          /* Class style word */
    FARPROC winClassProc;           /* Window function declared for class */
    HANDLE  winInstance;            /* Instance handle for window owner */
    HWND    winHWndParent;          /* Parent window handle */
    char    winTitle[TITLEMAX];     /* Window title or content string */
    WORD    winControlID;           /* Control ID or menu handle */
    FARPROC winWndProc;             /* Window function, usually = class fun. */
    DWORD   winStyle;               /* Style doubleword for window (WS_...) */
    RECT    winWindowRect;          /* Window rectangle (screen-relative) */
    RECT    winClientRect;          /* Client rectangle within window rect. */
} INFO;

typedef HANDLE      HINFO;          /* Handle to array of INFO structures */
typedef INFO huge * LPINFO;         /* Far pointer to same when locked down */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  The CsrScroll array is used for implementing keyboard scrolling.  By
 *  looking up the keystroke in this array, we get the equivalent scroll
 *  bar message.
 */

#define VK_MIN_CURSOR  VK_PRIOR
#define VK_MAX_CURSOR  VK_DOWN

struct {
    char    csBar;      /* Which scroll bar this key is equivalent to */
    char    csMsg;      /* The scroll message for this key */
} CsrScroll[] = {
    { SB_VERT, SB_PAGEUP   },  /* VK_PRIOR (PgUp)          */
    { SB_VERT, SB_PAGEDOWN },  /* VK_NEXT  (PgDn)          */
    { SB_VERT, SB_BOTTOM   },  /* VK_END   (End)           */
```

continued

```
    { SB_VERT, SB_TOP      },  /* VK_HOME  (Home)        */
    { SB_HORZ, SB_LINEUP   },  /* VK_LEFT  (left arrow)  */
    { SB_VERT, SB_LINEUP   },  /* VK_UP    (up arrow)    */
    { SB_HORZ, SB_LINEDOWN },  /* VK_RIGHT (right arrow) */
    { SB_VERT, SB_LINEDOWN }   /* VK_DOWN  (down arrow)  */
);

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Static variables
 */

HANDLE      hInstance;              /* Our instance handle */
HINFO       hInfo;                  /* Global handle to INFO array structure */
LPINFO      lpInfo;                 /* Far pointer to INFO, when locked down */
int         nWindows;               /* Total number of windows in system */
DWORD       dwInfoSize;             /* Size of entire INFO array in bytes */
FARPROC     lpprocCountWindow;      /* ProcInstance for CountWindow */
FARPROC     lpprocSpyOnWindow;      /* ProcInstance for SpyOnWindow */
BOOL        bInitted = FALSE;       /* TRUE when initialization completed */
BOOL        bExpand = FALSE;        /* Expanded display mode? */
int         nLinesPerWindow = 1;    /* 1 or LINES_PER_WINDOW */
int         nCharSizeX;             /* Width of a character in pixels */
int         nCharSizeY;             /* Height of a character in pixels */
int         nExtLeading;            /* # pixels vertical space between chars */
int         nPaintX;                /* For Paint function: X coordinate */
int         nPaintY;                /* For Paint function: Y coordinate */
HDC         hdcPaint;               /* For Paint function: hDC to paint into */
char        szClass[10];            /* Our window class name */
char        szTitle[40];            /* Our window title */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Declare full templates for all our functions.  This gives us strong type
 *  checking on our functions.
 */

BOOL    FAR PASCAL  CountWindow( HWND, long );
int                 DoScrollMsg( HWND, int, WORD, int );
void                HomeScrollBars( HWND, BOOL );
BOOL                Initialize( HANDLE, int );
void        cdecl   Paint( char *, ... );
void                PaintWindow( HWND );
void                SetScrollBars( HWND );
void                SetScrollBar1( HWND, int, int );
BOOL                SpyOnAllWindows( HWND );
BOOL    FAR PASCAL  SpyOnWindow( HWND, long );
long    FAR PASCAL  SpyWndProc( HWND, unsigned, WORD, LONG );
int         PASCAL  WinMain( HANDLE, HANDLE, LPSTR, int );

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Enumeration function to count the number of windows in the system.  Called
 *  once for each window, via EnumWindows and recursively via EnumChildWindows.
 *  The lTopLevel parameter tells us which kind of call it is.
 */

BOOL FAR PASCAL CountWindow( hWnd, lTopLevel )
    HWND        hWnd;               /* Window handle for this window */
    long        lTopLevel;          /* 1=top level window, 0=child window */
{
    /* Count the window */
    dwInfoSize += sizeof(INFO);
    ++nWindows;

    /* If this is a top level window (or popup), count its children */
    if( lTopLevel )  EnumChildWindows( hWnd, lpprocCountWindow, 0L );

    return TRUE;  /* TRUE to continue enumeration */
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Process a scroll bar message.  Calculates the distance to scroll based on
 *  the scroll bar range and the message code.  Limits the scroll to the actual
 *  range of the scroll bar.  Sets the new scroll bar thumb position and
 *  scrolls the window by the necessary amount.  Note that the scroll bar
 *  ranges are set in terms of number of characters, while the window scrolling
 *  is done by a number of pixels.  Returns the distance scrolled in chars.
 */

int DoScrollMsg( hWnd, nBar, wCode, nThumb )
    HWND        hWnd;               /* Window handle to scroll */
    int         nBar;               /* Which scroll bar: SB_HORZ or SB_VERT */
    WORD        wCode;              /* The scroll bar message code */
    int         nThumb;             /* Thumb position for SB_THUMBPOSITION */
```

```
{
    int         nOld;               /* Previous scroll bar position */
    int         nDiff;              /* Amount to change scroll bar by */
    int         nMin;               /* Minimum value of scroll bar range */
    int         nMax;               /* Maximum value of scroll bar range */
    int         nPageSize;          /* Size of our window in characters */
    RECT        rect;               /* Client rectangle for our window */

    /* Get old scroll position and scroll range */
    nOld = GetScrollPos( hWnd, nBar );
    GetScrollRange( hWnd, nBar, &nMin, &nMax );

    /* Quit if there is nowhere to scroll to (see SetScrollBars) */
    if( nMax == MAXINT )  return 0;

    /* Calculate page size, horizontal or vertical as needed */
    GetClientRect( hWnd, &rect );
    if( nBar == SB_HORZ )  nPageSize = (rect.right - rect.left) / nCharSizeX;
    else                   nPageSize = (rect.bottom - rect.top) / nCharSizeY;

    /* Select the amount to scroll by, based on the scroll message */
    switch( wCode ) {

        case SB_LINEUP:
            nDiff = -1;
            break;

        case SB_LINEDOWN:
            nDiff = 1;
            break;

        case SB_PAGEUP:
            nDiff = -nPageSize;
            break;

        case SB_PAGEDOWN:
            nDiff = nPageSize;
            break;

        case SB_THUMBPOSITION:
            nDiff = nThumb - nOld;
            break;

        case SB_TOP:
            nDiff = -30000;  /* Kind of a kludge but it works... */
            break;

        case SB_BOTTOM:
            nDiff = 30000;
            break;

        default:
            return 0;
    }

    /* Limit scroll destination to nMin..nMax */
    if( nDiff < nMin - nOld )  nDiff = nMin - nOld;
    if( nDiff > nMax - nOld )  nDiff = nMax - nOld;

    if( nDiff == 0 )  return 0;  /* Return if net effect is nothing */

    /* OK, now we can set the new scroll bar position and scroll the window */
    SetScrollPos( hWnd, nBar, nOld + nDiff, TRUE );

    ScrollWindow(
        hWnd,
        nBar == SB_HORZ ?  -nDiff*nCharSizeX : 0,
        nBar == SB_HORZ ?  0 : -nDiff*nCharSizeY,
        NULL,
        NULL
    );

    /* Force an immediate update for cleaner appearance */
    UpdateWindow( hWnd );

    return nDiff;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Set both scroll bars to the home position (0)
 */
```

continued

```
void HomeScrollBars( hWnd, bRedraw )
    HWND        hWnd;               /* Window handle */
    BOOL        bRedraw;            /* TRUE if scroll bars should be redrawn */
{
    SetScrollPos( hWnd, SB_HORZ, 0, bRedraw );
    SetScrollPos( hWnd, SB_VERT, 0, bRedraw );
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Initialize the application.  Some of the initialization is different
 *  depending on whether this is the first instance or a subsequent instance.
 *  For example, we register our window class only in the first instance.
 *  Returns TRUE if initialization succeeded, FALSE if failed.
 */

BOOL Initialize( hPrevInst, nCmdShow )
    HANDLE      hPrevInst;          /* Previous instance handle, 0 if first */
    int         nCmdShow;           /* Parameter from WinMain for ShowWindow */
{
    WNDCLASS    Class;              /* Class structure for RegisterClass */
    HWND        hWnd;               /* Our window handle */
    HDC         hDC;                /* Display context for our window */
    TEXTMETRIC  Metrics;            /* Text metrics for System font */
    HMENU       hMenu;              /* Menu handle of system menu */
    int         nScreenX;
    int         nScreenY;

    nScreenX = GetSystemMetrics( SM_CXSCREEN );
    nScreenY = GetSystemMetrics( SM_CYSCREEN );

    if( ! hPrevInst ) (
        /* Initialization for first instance only */

        /* Load strings from resource file */
        LoadString( hInstance, IDS_CLASS,    szClass,    sizeof(szClass) );
        LoadString( hInstance, IDS_TITLE,    szTitle,    sizeof(szTitle) );

        /* Register our window class */
        Class.style          = CS_HREDRAW | CS_VREDRAW;
        Class.lpfnWndProc    = SpyWndProc;
        Class.cbClsExtra     = 0;
        Class.cbWndExtra     = 0;
        Class.hInstance      = hInstance;
        Class.hIcon          = LoadIcon( hInstance, szClass );
        Class.hCursor        = LoadCursor( NULL, IDC_ARROW );
        Class.hbrBackground  = COLOR_WINDOW + 1;
        Class.lpszMenuName   = szClass;
        Class.lpszClassName  = szClass;

        if( ! RegisterClass( &Class ) )  return FALSE;

    ) else (
        /* Initialization for subsequent instances only */

        /* Copy data from previous instance */
        GetInstanceData( hPrevInst, szClass,    sizeof(szClass) );
        GetInstanceData( hPrevInst, szTitle,    sizeof(szTitle) );
    )

    /* Initialization for every instance */

    /* Set up ProcInstance pointers for our Enumerate functions */
    lpprocCountWindow = MakeProcInstance( CountWindow, hInstance );
    lpprocSpyOnWindow = MakeProcInstance( SpyOnWindow, hInstance );
    if( ! lpprocCountWindow || ! lpprocSpyOnWindow )  return FALSE;

    /* Allocate our INFO structure with nothing really allocated yet */
    hInfo = GlobalAlloc( GMEM_MOVEABLE, 1L );
    if( ! hInfo )  return FALSE;

    /* Create our tiled window but don't display it yet */
    hWnd = CreateWindow(
        szClass,                    /* Class name */
        szTitle,                    /* Window title */
        WS_TILEDWINDOW | WS_HSCROLL | WS_VSCROLL,  /* Window style */
        nScreenX *  1 / 20,         /* X: 5% from left */
        nScreenY *  1 / 10,         /* Y  10% from top */
        nScreenX *  9 / 10,         /* nWidth: 90% */
        nScreenY *  7 / 10,         /* nHeight: 70% */
        NULL,                       /* Parent hWnd (none for top-level) */
        NULL,                       /* Menu handle */
        hInstance,                  /* Owning instance handle */
        NULL                        /* Parameter to pass in WM_CREATE (none) */
```

```
    );

    /* Initialize scroll bars - Windows doesn't do this for us */
    HomeScrollBars( hWnd, FALSE );

    /* Calculate character size for system font */
    hDC = GetDC( hWnd );
    GetTextMetrics( hDC, &Metrics );
    ReleaseDC( hWnd, hDC );
    nExtLeading = Metrics.tmExternalLeading;
    nCharSizeX = Metrics.tmMaxCharWidth;
    nCharSizeY = Metrics.tmHeight + Metrics.tmExternalLeading;

    /* Make the window visible before grabbing spy info, so it's included */
    ShowWindow( hWnd, nCmdShow );

    /* Now grab the spy information */
    if( ! SpyOnAllWindows( hWnd ) )  return FALSE;

    /* Got all the information, update our display */
    UpdateWindow( hWnd );

    /* Make note that initialization is complete.  This is checked in our
     * routine that handles WM_SIZE to eliminate some jitter on startup */
    bInitted = TRUE;
    return TRUE;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Format and paint a line of text.  szFormat and Args are just as in a
 *  sprintf() call (Args is a variable number of arguments).  The global
 *  variables nPaintX and nPaintY tell where to paint the line.  We increment
 *  nPaintY to the next line after painting.
 *  Note the 'cdecl' declaration.  This forces this function to use the
 *  standard C calling sequence, which is necessary with a variable number
 *  of parameters.
 */

void cdecl Paint( szFormat, Args )
    char *      szFormat;           /* Format string as used in printf() */
    char        Args;               /* Zero or more parameters, as in printf */
{
    int         nLength;            /* Length of formatted string */
    char        Buf[160];           /* Buffer to format string into */

    nLength = vsprintf( Buf, szFormat, &Args );

    TextOut( hdcPaint, nPaintX, nPaintY+nExtLeading, Buf, nLength );
    nPaintY += nCharSizeY;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Paints our window or any portion of it that needs painting.
 *  The BeginPaint call sets up a structure that tells us what rectangle of the
 *  window to paint, along with other information for the painting process.
 *  First, erase the background area if necessary.
 *  Then, calculate the index into the INFO array to start with, based on the
 *  painting rectangle and the scroll bar position, and lock down the INFO.
 *  Finally, loop through the INFO array, painting the text for each entry.
 *  Quit when we run out of entries or hit the bottom of the paint rectangle.
 */

void PaintWindow( hWnd )
    HWND        hWnd;               /* Window handle to paint */
{
    PAINTSTRUCT ps;                 /* Paint structure set up by BeginPaint */
    DWORD       rgbOldTextColor;    /* Old text color (so we can restore it) */
    DWORD       rgbOldBkColor;      /* Old background color */
    int         nWin;               /* Index into INFO array */
    int         X;                  /* X position for paint calculation */
    int         Y;                  /* Y position for paint calculation */
    PSTR        pTypeName;          /* Pointer to "Child", etc. string */

    /* Tell Windows we're painting, set up the paint structure. */
    BeginPaint( hWnd, &ps );

    /* Store display context in global for Paint function */
    hdcPaint = ps.hdc;

    /* Set up proper background and text colors and save old values */
    rgbOldBkColor = SetBkColor( ps.hdc, GetSysColor( COLOR_WINDOW ) );
```

continued

```
rgbOldTextColor = SetTextColor( ps.hdc, GetSysColor( COLOR_WINDOWTEXT ) );
/* Calculate horizontal paint position based on scroll bar position */
X = ( 1 - GetScrollPos( hWnd, SB_HORZ ) ) * nCharSizeX;

/* Calculate index into INFO array and vertical paint position, based on
 * scroll bar position and top of painting rectangle */
Y = GetScrollPos( hWnd, SB_VERT );
nWin = ( ps.rcPaint.top / nCharSizeY + Y ) / nLinesPerWindow;
nPaintY = ( nWin * nLinesPerWindow - Y ) * nCharSizeY;

/* Lock down INFO array and set lpInfo pointing to first entry to paint */
lpInfo = (LPINFO)GlobalLock( hInfo );
lpInfo += nWin;

/* Loop through INFO entries, painting each one until we run out of entries
 * or until we are past the bottom of the paint rectangle.  We don't worry
 * much about painting outside the rectangle - Windows will clip for us. */
while( nWin < nWindows  &&  nPaintY < ps.rcPaint.bottom )
{
    /* Set X position and indent child windows, also set up pTypeName */
    nPaintX = X;
    if( lpInfo->winStyle & WS_CHILD ) {
        nPaintX += nCharSizeX * ( bExpand ? 4 : 2 );
        pTypeName = "Child";
    } else if( lpInfo->winStyle & WS_ICONIC ) {
        pTypeName = "Icon ";
    } else if( lpInfo->winStyle & WS_POPUP ) {
        pTypeName = "Popup";
    } else {
        pTypeName = "Top Level";
    }

    if( ! bExpand ) {

        /* Paint the one-liner */
        Paint(
            "%s window %04X (%Fs) (%d,%d;%d,%d) \"%Fs\"",
            pTypeName,
            lpInfo->winHWnd,
            lpInfo->winClass,
            lpInfo->winWindowRect.left,
            lpInfo->winWindowRect.top,
            lpInfo->winWindowRect.right,
            lpInfo->winWindowRect.bottom,
            lpInfo->winTitle
        );

    } else {

        /* Paint the expanded form, first the window handle */
        Paint(
            "%s window handle: %04X",
            pTypeName,
            lpInfo->winHWnd
        );

        /* Paint the rest of the info, indented two spaces farther over */
        nPaintX += nCharSizeX * 2;

        Paint( "Class name: %Fs", lpInfo->winClass );
        Paint( "Window title: %Fs", lpInfo->winTitle );
        Paint( "Parent window handle: %04X", lpInfo->winHWndParent );
        Paint(
            "Class function, Window function: %p, %p",
            lpInfo->winClassProc,
            lpInfo->winWndProc
        );
        Paint(
            "Class module handle, Window instance handle: %04X, %04X",
            lpInfo->winClassModule,
            lpInfo->winInstance
        );
        Paint(
            "Class extra alloc, Window extra alloc: %d, %d",
            lpInfo->winClsExtra,
            lpInfo->winWndExtra
        );
        Paint(
            "Class style, Window style: %04X, %081X",
            lpInfo->winClassStyle,
            lpInfo->winStyle
        );
        Paint(
            lpInfo->winStyle & WS_CHILD ?  "Control ID: %d" :
                                           "Menu handle: %04X",
```

```
                lpInfo->winControlID
            );
            Paint(
                "Brush, Cursor, Icon handles: %04X, %04X, %04X",
                lpInfo->winBkgdBrush,
                lpInfo->winCursor,
                lpInfo->winIcon
            );
            Paint(
                "Window rectangle: Left=%4d, Top=%4d, Right=%4d, Bottom=%4d",
                lpInfo->winWindowRect.left,
                lpInfo->winWindowRect.top,
                lpInfo->winWindowRect.right,
                lpInfo->winWindowRect.bottom
            );
            Paint(
                "Client rectangle: Left=%4d, Top=%4d, Right=%4d, Bottom=%4d",
                lpInfo->winClientRect.left,
                lpInfo->winClientRect.top,
                lpInfo->winClientRect.right,
                lpInfo->winClientRect.bottom
            );

            /* Make a blank line - it's already erased, so just increment Y */
            nPaintY += nCharSizeY;
        )

        /* Increment to next INFO entry */
        ++nWin;
        ++lpInfo;
    )

    /* Unlock the INFO array */
    GlobalUnlock( hInfo );

    /* Restore old colors */
    SetBkColor( ps.hdc, rgbOldBkColor );
    SetTextColor( ps.hdc, rgbOldTextColor );

    /* Tell Windows we're done painting */
    EndPaint( hWnd, &ps );
)

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Set horizontal and vertical scroll bars, based on the window size and the
 *  number of INFO entries.  The scroll bar ranges are set to give a total
 *  width of WINDOW_WIDTH and a total height equal to the number of lines of
 *  information available.  For example, if there are 130 lines of information
 *  and the window height is 10 characters, the vertical scroll range is set
 *  to 120 (130-10).  This lets you scroll through everything and still have
 *  a full window of information at the bottom.  (Unlike, say, Windows Write,
 *  where if you scroll to the bottom you have a blank screen.)
 */

void SetScrollBars( hWnd )
    HWND        hWnd;               /* Window handle */
(
    RECT        rect;               /* The window's client rectangle */

    GetClientRect( hWnd, &rect );

    SetScrollBar1(
        hWnd, SB_HORZ,
        WINDOW_WIDTH - rect.right / nCharSizeX
    );

    SetScrollBar1(
        hWnd, SB_VERT,
        nWindows * nLinesPerWindow - rect.bottom / nCharSizeY
    );
)

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Set one scroll bar's maximum range.  We always set the minimum to zero,
 *  although Windows allows other values.  There is one case we handle
 *  specially.  If you set a scroll bar range to minimum==maximum (maximum =
 *  zero for us), Windows does not actually set the range, but instead turns
 *  off the scroll bar completely, changing the window style by turning off
 *  the WS_HSCROLL or WS_VSCROLL bit.  For example, this is how the MS-DOS
 *  Executive makes its scroll bars appear and disappear.  This behavior is
 *  fine if you take it into account in your programming in two ways.  First,
 *  whenever you do a GetScrollRange you must first check the window style to
```

continued

```
 *  see if that scroll bar still exists, because you will *not* get the correct
 *  answer from GetScrollRange if it has been removed.  Second, you must be
 *  prepared to get some extra WM_SIZE messages, because your client area
 *  changes size when the scroll bars appear and disappear.  This can cause
 *  some sloppy looking screen painting.  We take a different approach, always
 *  keeping the scroll bars visible.  If the scroll bar range needs to be set
 *  to zero, instead we set it to MAXINT so the bar remains visible.  Then, in
 *  DoScrollMessage we check for this case and return without scrolling.
 */

void SetScrollBar1( hWnd, nBar, nMax )
    HWND        hWnd;               /* Window handle */
    int         nBar;               /* Which scroll bar: SB_HORZ or SB_VERT */
    int         nMax;               /* Value to set maximum range to */
{
    int         nOldMin;            /* Previous minimum value (always 0) */
    int         nOldMax;            /* Previous maximum value */

    /* Check for a negative or zero range and set our special case flag.
     * Also, set the thumb position to zero in this case. */
    if( nMax <= 0 ) {
        nMax = MAXINT;
        DoScrollMsg( hWnd, nBar, SB_THUMBPOSITION, 0 );
    }

    /* Find out the previous range, and set it if it has changed */
    GetScrollRange( hWnd, nBar, &nOldMin, &nOldMax );
    if( nMax != nOldMax )  SetScrollRange( hWnd, nBar, 0, nMax, TRUE );
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Loop through all windows in the system and gather up information for the
 *  INFO structure for each.  Use the EnumWindows and EnumChildWindows
 *  functions to loop through them.  We actually loop through them twice:
 *  first, to simply count them so we can allocate global memory for the
 *  INFO structure, and again to actually fill in the structure.  After
 *  gathering up the information, we invalidate our window, which will cause
 *  a WM_PAINT message to be posted, so it will get repainted.
 */

BOOL SpyOnAllWindows( hWnd )
    HWND        hWnd;               /* Window handle */
{
    /* Calculate the number of windows and amount of memory needed */
    nWindows = 0;
    dwInfoSize = 0;
    EnumWindows( lpprocCountWindow, 1L );

    /* Allocate the memory, complain if we couldn't get it */
    hInfo = GlobalReAlloc( hInfo, dwInfoSize, GMEM_MOVEABLE );
    if( ! hInfo ) {
        nWindows = 0;
        dwInfoSize = 0;
        GlobalDiscard( hInfo );
        MessageBox(
            GetActiveWindow(),
            "Insufficient memory!!",
            NULL,
            MB_OK | MB_ICONHAND
        );
        PostQuitMessage( 0 );
        return FALSE;
    }

    /* Lock down the memory and fill in the information, then unlock it */
    lpInfo = (LPINFO)GlobalLock( hInfo );
    EnumWindows( lpprocSpyOnWindow, 1L );
    GlobalUnlock( hInfo );

    /* Set the scroll bars based on new window count, repaint our window */
    SetScrollBars( hWnd );
    HomeScrollBars( hWnd, TRUE );
    InvalidateRect( hWnd, NULL, TRUE );

    return TRUE;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Enumeration function to gather up the information for a single window and
 *  store it in the INFO array entry pointed to by lpInfo.  Increment lpInfo
 *  to the next entry afterward.  Called once for each window, via EnumWindows
 *  for each top level and popup window, and recursively via EnumChildWindows
 *  for child windows.  The lTopLevel parameter tells which kind of call it is.
 */
```

```
BOOL FAR PASCAL SpyOnWindow( hWnd, lTopLevel )
    HWND        hWnd;               /* Window handle */
    long        lTopLevel;          /* 1=top level window, 0=child window */
{
    /* Gather up this window's information */
    lpInfo->winHWnd = hWnd;
    GetClassName( hWnd, lpInfo->winClass, CLASSMAX );
    lpInfo->winClass[ CLASSMAX - 1 ] = 0;
    lpInfo->winInstance = GetWindowWord( hWnd, GWW_HINSTANCE );
    lpInfo->winHWndParent = GetParent( hWnd );
    GetWindowText( hWnd, lpInfo->winTitle, TITLEMAX );
    lpInfo->winTitle[ TITLEMAX - 1 ] = 0;
    lpInfo->winControlID = GetWindowWord( hWnd, GWW_ID );
    lpInfo->winWndProc = (FARPROC)GetWindowLong( hWnd, GWL_WNDPROC );
    lpInfo->winStyle = GetWindowLong( hWnd, GWL_STYLE );
    GetClientRect( hWnd, &lpInfo->winClientRect );
    GetWindowRect( hWnd, &lpInfo->winWindowRect );

    /* Gather up class information */
    lpInfo->winBkgdBrush = GetClassWord( hWnd, GCW_HBRBACKGROUND );
    lpInfo->winCursor = GetClassWord( hWnd, GCW_HCURSOR );
    lpInfo->winIcon = GetClassWord( hWnd, GCW_HICON );
    lpInfo->winClassModule = GetClassWord( hWnd, GCW_HMODULE );
    lpInfo->winWndExtra = GetClassWord( hWnd, GCW_CBWNDEXTRA );
    lpInfo->winClsExtra = GetClassWord( hWnd, GCW_CBCLSEXTRA );
    lpInfo->winClassStyle = GetClassWord( hWnd, GCW_STYLE );
    lpInfo->winClassProc = (FARPROC)GetClassLong( hWnd, GCL_WNDPROC );

    /* Move on to next entry in table */
    ++lpInfo;

    /* If it's a top level window, get its children too */
    if( lTopLevel )  EnumChildWindows( hWnd, lpprocSpyOnWindow, 0L );

    return TRUE;  /* TRUE to continue enumeration */
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Window function for our main window.  All messages for our window are sent
 *  to this function.  For messages that we do not handle here, we call
 *  DefWindowProc, which performs Windows' default processing for a message.
 */

long FAR PASCAL SpyWndProc( hWnd, wMsg, wParam, lParam )
    HWND        hWnd;               /* Window handle */
    unsigned    wMsg;               /* Message number */
    WORD        wParam;             /* Word parameter for the message */
    LONG        lParam;             /* Long parameter for the message */
{
    RECT        rect;               /* A rectangle */

    switch( wMsg ) {

        /* Menu command message - process the command */
        case WM_COMMAND:
            if( LOWORD(lParam) )  break;  /* not a command */
            switch( wParam ) {
                case CMD_EXPAND:
                    bExpand = ! bExpand;
                    nLinesPerWindow = ( bExpand ? LINES_PER_WINDOW : 1 );
                    CheckMenuItem(
                        GetMenu( hWnd ),
                        CMD_EXPAND,
                        bExpand ? MF_CHECKED : MF_UNCHECKED
                    );
                    InvalidateRect( hWnd, NULL, TRUE );
                    HomeScrollBars( hWnd, FALSE );
                    SetScrollBars( hWnd );
                    return 0L;
                case CMD_SPY:
                    SpyOnAllWindows( hWnd );
                    return 0L;
                default:
                    break;
            }
            break;

        /* Destroy-window message - time to quit the application */
        case WM_DESTROY:
            PostQuitMessage( 0 );
            return 0L;
```

continued

```
        /* Horizontal scroll message - scroll the window */
        case WM_HSCROLL:
            DoScrollMsg( hWnd, SB_HORZ, wParam, (int)lParam );
            return 0L;

        /* Key-down message - handle cursor keys, ignore other keys */
        case WM_KEYDOWN:
            if( wParam >= VK_MIN_CURSOR  &&  wParam <= VK_MAX_CURSOR ) {
                DoScrollMsg(
                    hWnd,
                    CsrScroll[ wParam - VK_MIN_CURSOR ].csBar,
                    CsrScroll[ wParam - VK_MIN_CURSOR ].csMsg,
                    0
                );
            }
            return 0L;

        /* Paint message - repaint all or part of our window */
        case WM_PAINT:
            PaintWindow( hWnd );
            return 0L;

        /* Size message - recalculate our scroll bars to take the new size
         * into account, but only if initialization has been completed.  There
         * are several superfluous WM_SIZE messages sent during initialization,
         * and it looks ugly if we repaint the scroll bars for all these. */
        case WM_SIZE:
            if( bInitted )  SetScrollBars( hWnd );
            return 0L;

        /* Vertical scroll message - scroll the window */
        case WM_VSCROLL:
            DoScrollMsg( hWnd, SB_VERT, wParam, (int)lParam );
            return 0L;

        /* For all other messages, we pass them on to DefWindowProc */
        default:
            break;
    }
    return DefWindowProc( hWnd, wMsg, wParam, lParam );
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Application main program.  Not much is done here - we just initialize
 *  the application, putting up our window, and then we go into the typical
 *  message dispatching loop that every Windows application has.
 */

int PASCAL WinMain( hInst, hPrevInst, lpszCmdLine, nCmdShow )
    HANDLE      hInst;              /* Our instance handle */
    HANDLE      hPrevInst;          /* Previous instance of this application */
    LPSTR       lpszCmdLine;        /* Pointer to any command line params */
    int         nCmdShow;           /* Parameter to use for first ShowWindow */
{
    MSG         msg;                /* Message structure */

    /* Save our instance handle in static variable */
    hInstance = hInst;

    /* Initialize application, quit if any errors */
    if( ! Initialize( hPrevInst, nCmdShow ) )  return FALSE;

    /* Main message processing loop.  Get each message, then translate keyboard
     * messages, and finally dispatch each message to its window function. */
    while( GetMessage( &msg, NULL, 0, 0 ) ) {
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }

    return msg.wParam;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

SPY.DEF is from "Spying on Windows", by Mike Geary, BYTE, IBM Special Issue, 1987, page 97.

```
NAME    Spy
```

```
DESCRIPTION 'Windows Espionage'

STUB    'WINSTUB.EXE'

CODE    MOVEABLE
DATA    MOVEABLE MULTIPLE

HEAPSIZE    1024
STACKSIZE   4096

EXPORTS
    CountWindow     @1
    SpyOnWindow     @2
    SpyWndProc      @3
```

SPY.RC is from "Spying on Windows", by Mike Geary, BYTE, IBM Special Issue, 1987, page 97.

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Spy.rc - resource file for SPY.EXE */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

#include <style.h>
#include "spy.h"

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

Spy!    ICON    spy.ico

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

STRINGTABLE
BEGIN
    IDS_CLASS,      "Spy!"
    IDS_TITLE,      "Spy on Windows!"
END

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

Spy!    MENU
BEGIN

    POPUP   "&Spy"
    BEGIN
        MENUITEM "&New Spy Mission", CMD_SPY
        MENUITEM SEPARATOR
        MENUITEM "Show &Detail", CMD_EXPAND
    END

END

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

SPY.H is from "Spying on Windows", by Mike Geary, BYTE, IBM Special Issue, 1987, page 97.

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* SPY.H */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

#define MAXINT      32767
#define MAXWORD     65535

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Menu command definitions */
```

continued

```
#define CMD_SPY     1
#define CMD_EXPAND  2

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* String table ID numbers */

#define IDS_CLASS       1
#define IDS_TITLE       2

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

SPY is from "Spying on Windows", by Mike Geary, BYTE, IBM Special Issue, 1987, page 97.

```
# Makefile for SPY.EXE

spy.obj:    spy.c
    msc -AS -Gcsw -Os -u -W2 -Zdp $*;
    findwarn

spy.res:    spy.rc  spy.ico
    rc -r spy.rc

spy.exe:    spy.obj  spy.res  spy.def
    link4 spy, spy/align:16, spy/map/line, slibw, spy.def
    mapsym4 spy
    rc spy.res
```

SPY.DOC is from "Spying on Windows", by Mike Geary, BYTE, IBM Special Issue, 1987, page 97.

This is the SPY program from my article "Spying on Windows" in the 1987 IBM special issue of BYTE.

The source code included can be compiled with either the 1.03/4 or 2.0 Windows Software Development Kit. Although there are a few minor items in the SPY code that use Windows 2.0 features, nothing depends on them, so the source code is still compatible with the 1.0x development kit.

When Windows 2.0 becomes generally available, I will be coming out with a version of SPY that uses more of the Windows 2.0 facilities to do a little more thorough investigation of other applications. Stay tuned!

If you find SPY useful, drop me a note and let me know what interesting things you have done with it. (No, don't send money - it's a freebie!)

Michael Geary
P.O. Box 1479
Los Gatos, CA 95031

BIX: 'geary'
GEnie: GEARY
CompuServe: 76146,42

STAMPER.ASM is from "TSRs Past and Future: MS-DOS and OS/2", by Ray Duncan, BYTE, IBM Special Issue, 1987, page 49.

```
        name    stamper
        page    55,132
        title   STAMPER: a simple keyboard monitor
        .286c
        .sall
;
```

```
; STAMPER.EXE: a simple OS/2 monitor that inserts
; a date or time stamp into the keyboard data stream.
;
; Alt-D  is the hot-key for a date-stamp.
; Alt-T  is the hot-key for a time-stamp.
; Alt-X  causes the STAMPER.EXE monitor to exit.
;
; Copyright (C) 1987 Ray Duncan
;
; To assemble and link this program:
;
;       MASM stamper;
;       LINK stamper,,,DOSCALLS,stamper;
;
; To use this program, execute it with the command:
;
;       C>DETACH STAMPER
;

stdin   equ     0                       ; standard device handles
stdout  equ     1
stderr  equ     2

cr      equ     0dh                     ; ASCII carriage return
lf      equ     0ah                     ; ASCII line feed

                                        ; Hot-key definitions:
datekey equ     20h                     ; Alt-D
timekey equ     14h                     ; Alt-T
exitkey equ     2dh                     ; Alt-X

        extrn   DOSGETINFOSEG:far       ; OS/2 API services needed
        extrn   DOSEXIT:far
        extrn   DOSSLEEP:far
        extrn   DOSWRITE:far
        extrn   DOSMONOPEN:far
        extrn   DOSMONCLOSE:far
        extrn   DOSMONREG:far
        extrn   DOSMONREAD:far
        extrn   DOSMONWRITE:far
        extrn   VIOPOPUP:far
        extrn   VIOENDPOPUP:far
        extrn   VIOWRTCHARSTR:far

jerr    macro   target                  ; Macro to test return code
        local   zero                    ; in AX and jump if non-zero.
        or      ax,ax                   ; Uses JMP DISP16 to avoid
        jz      zero                    ; branch out of range errors
        jmp     target
zero:
        endm


DGROUP  group   _DATA

_DATA   segment word public 'DATA'

popflag dw      1                       ; wait for PopUp window
wlen    dw      ?                       ; receives length written

kname   db      '\DEV\KBD$',0           ; device name of keyboard
khandle dw      0                       ; handle from DosMonOpen

gseg    dw      ?                       ; global information segment
lseg    dw      ?                       ; local information segment

scrgrp  dw      ?                       ; foreground screen group

dstr    db      'mm/dd/yy',0            ; strings used by time and
tstr    db      'hh:mm',0               ; date formatting routines

monin   dw      128,64 dup (0)          ; buffers for monitor
monout  dw      128,64 dup (0)

packet  db      128 dup (0)             ; buffer for kbd data packet
pktlen  dw      ?                       ; length of buffer prior to read
                                        ; call, length of data after.

msg1    db      cr,lf,'Start STAMPER with DETACH!',cr,lf
msg1_len equ $-msg1

msg2    db      'STAMPER utility installed!'
msg2_len equ $-msg2
```

continued

```
msg3    db      'Alt-D to insert date stamp.'
msg3_len equ $-msg3

msg4    db      'Alt-T to insert time stamp.'
msg4_len equ $-msg4

msg5    db      'Alt-X to shut down STAMPER.'
msg5_len equ $-msg5

msg6    db      'STAMPER utility deactivated.'
msg6_len equ $-msg6

msg7    db      cr,lf,'Unexpected OS/2 error',cr,lf
msg7_len equ $-msg7

_DATA   ends


_TEXT   segment word public 'CODE'

        assume  cs:_TEXT,ds:DGROUP,ss:DGROUP

main    proc    far                     ; entry point from OS/2

        push    ds                      ; get selectors for system's
        push    offset DGROUP:gseg      ; global information segments
        push    ds
        push    offset DGROUP:lseg
        call    DOSGETINFOSEG           ; transfer to OS/2
        jerr    error                   ; give up if can't get selectors

        mov     es,lseg                 ; get our screen group number
        mov     ax,es:[8]               ; and make sure we are detached
        cmp     ax,16
        je      main1                   ; proceed, all is well

                                        ; not run with DETACH,
                                        ; display error message...
        push    stderr                  ; handle for standard error
        push    ds                      ; address of message
        push    offset DGROUP:msg1
        push    msg1_len                ; length of message
        jmp     error2                  ; go display and exit

main1:  mov     es,gseg                 ; get foreground screen group
        mov     al,byte ptr es:[0018h]  ; from global info segment
        cbw
        mov     scrgrp,ax

                                        ; open monitor connection...
        push    ds                      ; address of name \DEV\KBD$
        push    offset DGROUP:kname
        push    ds                      ; address to receive monitor handle
        push    offset DGROUP:khandle
        call    DOSMONOPEN              ; transfer to OS/2
        jerr    error                   ; give up if can't open it

                                        ; register as keyboard monitor...
        push    khandle                 ; handle from DosMonOpen
        push    ds                      ; addr of monitor input buffer
        push    offset DGROUP:monin
        push    ds                      ; addr of monitor output buffer
        push    offset DGROUP:monout
        push    1                       ; request front of list
        push    scrgrp                  ; screen group we are monitoring
        call    DOSMONREG               ; transfer to OS/2
        jerr    error                   ; give up if can't register

        call    signon                  ; else announce our presence

main2:                                  ; monitor the keyboard character
                                        ; stream; when hot key detected,
                                        ; insert the appropriate date or
                                        ; time stamp, or exit.

        mov     pktlen,pktlen-packet    ; set max buffer length for read

                                        ; get next keyboard data packet
        push    ds                      ; address of monitor input buffer
        push    offset DGROUP:monin
        push    0                       ; wait until data available
        push    ds
        push    offset DGROUP:packet    ; buffer for keyboard data packet
        push    ds
        push    offset DGROUP:pktlen    ; receives length of data packet
        call    DOSMONREAD              ; transfer to OS/2
```

```
        cmp     byte ptr packet+2,0     ; is this extended code?
        jnz     main4                   ; no, just pass it on

        cmp     byte ptr packet+3,exitkey
        jz      main5                   ; jump if exit hot-key

        cmp     byte ptr packet+3,timekey
        jnz     main3                   ; jump if not time hot-key

        cmp     word ptr packet+12,0    ; discard break packets
        jnz     main2

        call    time                    ; insert the time stamp

        jmp     main2                   ; discard this key

main3:                                  ; is it datestamp hot-key?
        cmp     byte ptr packet+3,datekey
        jnz     main4                   ; no, jump

        cmp     word ptr packet+12,0    ; discard break packets
        jnz     main2

        call    date                    ; insert the date stamp

        jmp     main2                   ; discard this key

main4:                                  ; Not hot-key, pass packet on.
        push    ds                      ; address of monitor output buffer
        push    offset DGROUP:monout
        push    ds                      ; address of keyboard data packet
        push    offset DGROUP:packet
        push    pktlen                  ; length of data packet
        call    DOSMONWRITE             ; transfer to OS/2

        jmp     main2                   ; wait for another packet

main5:                                  ; hotkey for de-install detected

        cmp     word ptr packet+12,0    ; make sure it's Break packet
        jz      main2                   ; if not just discard it

        push    khandle                 ; close the monitor connection
        call    DOSMONCLOSE
        jerr    error

        call    signoff                 ; announce STAMPER exit

        push    1                       ; terminate all threads
        push    0                       ; return success code
        call    DOSEXIT                 ; final exit to OS/2

main    endp

error   proc    near                    ; unilateral termination
                                        ; because of unexpected error.

        cmp     khandle,0               ; first shut down monitor
        je      error1                  ; if it is active
        push    khandle
        call    DOSMONCLOSE             ; ignore any error codes

error1:                                 ; write message 'Unexpected error'
        push    stderr                  ; handle for standard error device
        push    ds                      ; address of message
        push    offset DGROUP:msg7
        push    msg7_len                ; length of message

error2: push    ds                      ; receives bytes written
        push    offset DGROUP:wlen
        call    DOSWRITE                ; transfer to OS/2

        push    1                       ; terminate all threads
        push    1                       ; return error code
        call    DOSEXIT                 ; final exit to OS/2

error   endp


date    proc    near                    ; format and insert date stamp

        mov     es,gseg                 ; get selector for global
                                        ; read-only information segment
```

continued

```
        mov     al,byte ptr es:[11h]    ; convert month to ASCII
        aam
        add     ax,'00'
        xchg    al,ah
        mov     word ptr dstr,ax

        mov     al,byte ptr es:[10h]    ; convert day to ASCII
        aam
        add     ax,'00'
        xchg    al,ah
        mov     word ptr dstr+3,ax

        mov     ax,word ptr es:[12h]    ; convert year to ASCII
        sub     ax,1900
        aam
        add     ax,'00'
        xchg    al,ah
        mov     word ptr dstr+6,ax

        mov     si,offset DGROUP:dstr   ; insert date stamp string
        call    stuff                   ; into keyboard data stream

        ret                             ; back to caller

date    endp


time    proc    near                    ; format and insert time stamp

        mov     es,gseg                 ; get selector for global

                                        ; read-only information segment

        mov     al,byte ptr es:[8]      ; convert hours to ASCII
        aam
        add     ax,'00'
        xchg    al,ah
        mov     word ptr tstr,ax

        mov     al,byte ptr es:[9]      ; convert minutes to ASCII
        aam
        add     ax,'00'
        xchg    al,ah
        mov     word ptr tstr+3,ax

        mov     si,offset DGROUP:tstr   ; insert time stamp string
        call    stuff                   ; into keyboard data stream

        ret                             ; back to caller

time    endp


stuff   proc    near                    ; insert string into keyboard
                                        ; data stream.  Call with
                                        ; SI = ASCIIZ string (null
                                        ;      is discarded)
                                        ; AL, SI destroyed.

stuff1: lodsb                           ; get next character
        or      al,al                   ; is it null?
        jnz     stuff2                  ; no, use it
        ret                             ; yes, exit

stuff2: mov     packet+2,al             ; place ASCII code into packet

                                        ; now send this character
                                        ; to the keyboard driver...
        push    ds                      ; address of monitor output buffer
        push    offset DGROUP:monout
        push    ds                      ; address of keyboard data packet
        push    offset DGROUP:packet
        push    pktlen                  ; length of data packet
        call    DOSMONWRITE             ; transfer to OS/2
        jmp     stuff1                  ; do another character

stuff   endp


signon  proc    near                    ; use pop-up window to
                                        ; display help message

        push    ds                      ; put up PopUp window
        push    offset DGROUP:popflag   ; (wait until available)
        push    0
        call    VIOPOPUP
```

```
        mov     dx,offset DGROUP:msg2   ; message address
        mov     cx,msg2_len             ; length
        mov     ax,9                    ; Y coordinate
        call    center                  ; display it
        mov     dx,offset DGROUP:msg3   ; message address
        mov     cx,msg3_len             ; length
        mov     ax,13                   ; Y coordinate
        call    center                  ; display it

        mov     dx,offset DGROUP:msg4   ; message address
        mov     cx,msg4_len             ; length
        mov     ax,15                   ; Y coordinate
        call    center                  ; display it

        mov     dx,offset DGROUP:msg5   ; message address
        mov     cx,msg5_len             ; length
        mov     ax,17                   ; Y coordinate
        call    center                  ; display it

        push    0                       ; pause for 3 seconds
        push    3000                    ; (user must be quick reader!)
        call    DOSSLEEP

        push    0                       ; take down PopUp window
        call    VIOENDPOPUP

        ret                             ; back to caller

signon  endp


signoff proc    near                    ; use pop-up window to
                                        ; announce exit

        push    ds                      ; put up PopUp window
        push    offset DGROUP:popflag   ; (wait until available)
        push    0
        call    VIOPOPUP

        mov     dx,offset DGROUP:msg6   ; message address
        mov     cx,msg6_len             ; length
        mov     ax,12                   ; Y coordinate
        call    center                  ; display it

        push    0                       ; pause for 1 seconds
        push    1000                    ; (user must not blink
        call    DOSSLEEP                ;  at wrong time...)

        push    0                       ; take down PopUp window
        call    VIOENDPOPUP

        ret                             ; back to caller

signoff endp


center  proc    near                    ; center a message on screen
                                        ; call DX = msg offset,
                                        ; CX = length, AX = Y coordinate

        push    ds                      ; address of message
        push    dx
        push    cx                      ; length of message
        push    ax                      ; Y
        sub     cx,80                   ; X=((80-length)/2)
        neg     cx
        shr     cx,1
        push    cx
        push    0                       ; VIO handle
        call    VIOWRTCHARSTR           ; transfer to OS/2
        ret                             ; back to caller

center  endp

_TEXT   ends

        end     main
```

continued

STAMPER.DEF is from "TSRs Past and Future: MS-DOS and OS/2", by
Ray Duncan, BYTE, IBM Special Issue, 1987, page 49.

```
------------------------------------------------------------------------
The module definition file:  STAMPER.DEF
------------------------------------------------------------------------

NAME STAMPER
PROTMODE
DATA MOVEABLE
CODE MOVEABLE PURE
STACKSIZE 4096

------------------------------------------------------------------------
```

STAMPER.MAK is from "TSRs Past and Future: MS-DOS and OS/2", by
Ray Duncan, BYTE, IBM Special Issue, 1987, page 49.

```
------------------------------------------------------------------------
The MAKE file:  STAMPER
------------------------------------------------------------------------

stamper.obj : stamper.asm
 masm /Zi /T stamper;

stamper.exe : stamper.obj stamper.def stamper
 link /CO /MAP stamper,,,doscalls,stamper

------------------------------------------------------------------------
The End!
------------------------------------------------------------------------
```

STRING.ASM is from "Better Batch Files Through Assembler", by
William J. Claff, BYTE, IBM Special Issue, 1987, page 159.

```
VECTOR  STRUC
REGIP   DW      ?
REGCS   DW      ?
VECTOR  ENDS
CODE    SEGMENT
        ASSUME  CS:CODE         ;<- ASSUMES FOR .COM FILE
        ASSUME  DS:CODE
        ASSUME  ES:CODE
        ASSUME  SS:CODE
        ORG     00100H          ;<- REQUIRED FOR .COM FILE
IP      LABEL   NEAR            ;   (USED ON END STATEMENT)
        JMP     SHORT START
HOLDS   DB      16              ;<- FOR DOS BUFFERED INPUT FUNCTION
HAS     DB      0
INPUT   DB      15 DUP(' '),0
SET     DB      11              ;<- FOR INT 2E
        DB      'SET STRING='
SETTING DB      16 DUP(?)
;*******INT 2E PROCEDURE
SS_SP   VECTOR  <>
INT2E   PROC    NEAR
        PUSH    AX              ;<- SAVE ALL REGISTERS
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    BP
        PUSH    SI
        PUSH    DI
        PUSH    DS
        PUSH    ES
        PUSHF
        MOV     SS_SP.REGIP,SP  ;<- SAVE SS:SP
        MOV     SS_SP.REGCS,SS
        INT     02EH            ;<- DO INT 2E
        ASSUME  DS:NOTHING
```

```
        ASSUME  ES:NOTHING
        ASSUME  SS:NOTHING
        MOV     SS,SS_SP.REGCS  ;<- RESTORE SS:SP
        ASSUME  SS:CODE
        MOV     SP,SS_SP.REGIP
        POPF                    ;<- RESTORE ALL REGISTERS
        POP     ES
        ASSUME  ES:CODE
        POP     DS
        ASSUME  DS:CODE
        POP     DI
        POP     SI
        POP     BP
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
INT2E   ENDP
START   LABEL   NEAR
        MOV     AH,04AH
        MOV     BX,OFFSET FREE
        DEC     BX
        MOV     CL,4
        SHR     BX,CL
        INC     BX
        INT     021H
        MOV     AH,00CH         ;<- GET INPUT STRING
        MOV     AL,00AH
        MOV     DX,OFFSET HOLDS
        INT     021H
        XOR     CH,CH
        MOV     CL,HAS
        JCXZ    ERROR1
        ADD     SET,CL
        MOV     SI,OFFSET INPUT
        MOV     DI,OFFSET SETTING
        REP     MOVSB
        XOR     BH,BH
        MOV     BL,SET
        MOV     SET[BX][1],00DH
        MOV     SI,OFFSET SET
        CALL    INT2E
EXIT    LABEL   NEAR            ;<- EXIT
        MOV     AH,04CH
        INT     021H
ERROR1  LABEL   NEAR
        MOV     AL,1
        JMP     EXIT
FREE    LABEL   BYTE
CODE    ENDS
        END     IP              ;<- REQUIRED FOR .COM FILE
```

T6821.PAS is from "Three Bus Interface Designs for the PC", by James R. Drummond, BYTE, IBM Special Issue, 1987, page 225.

```
program test_6821;
const
     PORT_LOC = $388;
     CONTROL_LOC = $38A;
var
   i,j,k:  integer;
begin
     portw[PORT_LOC] := $ffff;
     port[CONTROL_LOC] := $0404;
     repeat
          begin
               for j := 0 to maxint do
               begin
                    portw[PORT_LOC] := j;
                    k := portw[PORT_LOC];
                    if k <> j then writeln('help'^G);
               end;
          end;
          writeln(k);
     until false;
end.
```

continued

T810.PAS is from "Three Bus Interface Designs for the PC", by
James R. Drummond, BYTE, IBM Special Issue, 1987, page 225.

```
program test_810;
const
     PORT_LOC = $380;
     NSC_0    = $000;
     MDR      = $7;
     PORT_AB  = 0;
     DDR_AB   = $4;

var
   i,j,k:          integer;

begin
     portw[PORT_LOC or NSC_0 or MDR] := 0;
     portw[PORT_LOC or NSC_0 or DDR_AB] := $ffff;
     repeat
          begin
               for j := 0 to maxint do
               begin
                    portw[PORT_LOC or NSC_0 or PORT_AB] := j;
                    k := portw[PORT_LOC or NSC_0 or PORT_AB];
                    if j <> k then writeln('help'^G);
               end;
          end;
          writeln(k);
     until false;
end.
```

T8255.PAS is from "Three Bus Interface Designs for the PC", by
James R. Drummond, BYTE, IBM Special Issue, 1987, page 225.

```
program test_8255;
const
     PORT_A       = $388;
     PORT_B       = $389;
     PORT_C       = $38A;
     CONTROL_LOC = $38B;
var
   i,j,k:  integer;
begin
     port[CONTROL_LOC] := $84;
     repeat
          begin
               for j := 0 to maxint do
               begin
                    port[PORT_A] := lo(j);
                    port[PORT_B] := hi(j);
                    i := port[PORT_A];
                    k := port[PORT_B];
                    if (hi(j) <> k) or (lo(j) <> i) then writeln('help'^G);
               end;
          end;
          writeln(k:6,i:6);
     until false;
end.
```

TOPATH.ASM is from "Better Batch Files Through Assembler", by
William J. Claff, BYTE, IBM Special Issue, 1987, page 159.

```
CODE     SEGMENT
         ASSUME  CS:CODE          ;<- ASSUMES FOR .COM FILE
         ASSUME  DS:CODE
         ASSUME  ES:CODE
         ASSUME  SS:CODE
         ORG     00080H           ;<- LENGTH OF COMMAND TAIL
TAILLEN  DB      ?
         ORG     00081H           ;<- COMMAND TAIL
TAIL     DB      127 DUP(?)
```

```
        ORG     00100H          ;<- REQUIRED FOR .COM FILE
IP      LABEL   NEAR            ;   (USED ON END STATEMENT)
        XOR     BH,BH           ;<- PUT NUL AT END OF COMMAND TAIL
        MOV     BL,TAILLEN
        MOV     TAIL[BX],000H
        MOV     AL,' '          ;<- SCAN FORWARD ACROSS BLANKS
        MOV     CX,BX
        MOV     DI,OFFSET TAIL
        CLD
        REPE    SCAS    TAIL[DI]
        DEC     DI
        MOV     AH,03BH         ;<- CHANGE DIRECTORY
        MOV     DX,DI
        INT     021H
        JC      EXIT
        XOR     AL,AL
EXIT    LABEL   NEAR            ;<- EXIT
        MOV     AH,04CH         ;   0=OK OTHERWISE=ERROR NUMBER
        INT     021H
CODE    ENDS
        END     IP              ;<- REQUIRED FOR .COM FILE
```

WINDOW.BAS is from "Windows for BASIC", by John W. Ross, BYTE, IBM Special Issue, 1987, page 201.

```
100 ' WINDOW.BAS - program to generate WINDOW.EXE
110 '
120 '  Copyright (C)  John W. Ross   1986
130 '
140 OPEN"window.exe" AS #2 LEN=1
150 FIELD #2, 1 AS Z$
160 READ A$
170 WHILE A$<>"-1"
180   C$="&h"+A$
190   C=VAL(C$)
200   O$=MKI$(C)
210   LSET Z$=LEFT$(O$,1)
220   PUT#2
230   READ A$
240 WEND
1000 DATA 4D, 5A, 40, 00, 03, 00, 02, 00, 20, 00, 00, 00, FF, FF, 18, 00, A0
1010 DATA 00, 12, C6, 00, 00, 22, 00, 1E, 00, 00, 00, 01, 00, 79, 00, 00, 00
1020 DATA 1A, 00, 22, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1030 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1040 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1050 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1060 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1070 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1080 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1090 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1100 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1110 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1120 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1130 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1140 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1150 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1160 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1170 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1180 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1190 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1200 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1210 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1220 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1230 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1240 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1250 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1260 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1270 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1280 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1290 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
1300 DATA 00, 00, 55, 8B, EC, 1E, 06, 16, EB, 39, 90, 43, 6F, 70, 79, 72, 69
1310 DATA 67, 68, 74, 20, 28, 43, 29, 20, 4A, 6F, 68, 6E, 20, 57, 2E, 20, 52
1320 DATA 6F, 73, 73, 20, 31, 39, 38, 36, 00, 00, 00, 00, 00, 00, 00, 00, 00
1330 DATA 00, 00, 00, 00, 00, 00, 00, 00, B8, 00, B9, 00, BA, 00, BB, 00, B0, 8B
1340 DATA 5E, 06, 8B, 07, 2E, A2, 34, 00, 8B, 5E, 08, 8B, 07, 2E, A3, 2C, 00
1350 DATA 8B, 5E, 0A, 8B, 07, 2E, A3, 2E, 00, 8B, 5E, 0C, 8B, 07, 2E, A3, 28
1360 DATA 00, 8B, 5E, 0E, 8B, 07, 2E, A3, 2A, 00, 8B, 5E, 10, 8B, 07, 2E, A3
1370 DATA 35, 00, 1E, B8, 00, 00, 8E, D8, B4, 0F, CD, 10, 3C, 02, 74, 0C, 3C
```

continued

```
1380 DATA 03, 74, 08, 3C, 07, 74, 04, 1F, E9, BC, 00, B4, 00, A3, 30, 00, 8A
1390 DATA DF, B7, 00, 89, 1E, 32, 00, D1, E3, 83, 3E, 30, 00, 07, 75, 03, BB
1400 DATA 08, 00, 8B, 97, 37, 00, 52, 8A, 36, 2A, 00, 8A, 16, 28, 00, B0, 50
1410 DATA F6, E6, B6, 00, 03, C2, D1, E0, 8B, D8, 07, 8A, 16, 2C, 00, 8A, 36
1420 DATA 2E, 00, A1, 35, 00, 1F, 8B, 7E, 12, 53, 8A, CE, B5, 00, 51, 53, 8A
1430 DATA CA, B5, 00, 51, 3D, 01, 00, 74, 08, 8B, 0D, 26, 89, 0F, EB, 06, 90
1440 DATA 26, 8B, 0F, 89, 0D, 43, 43, 47, 47, 59, E2, E6, 5B, 81, C3, A0, 00
1450 DATA 59, E2, D8, 0E, 1F, 5B, 3D, 01, 00, 75, 45, B2, C9, B6, BB, B0, CD
1460 DATA 8A, 26, 34, 00, 8B, 0E, 2C, 00, E8, 3B, 00, 81, C3, A0, 00, B2, BA
1470 DATA B6, BA, B0, 20, 8A, 26, 34, 00, 8B, 0E, 2E, 00, 49, 49, 51, 8B, 0E
1480 DATA 2C, 00, E8, 1F, 00, 81, C3, A0, 00, 59, E2, F1, B2, C8, B6, BC, B0
1490 DATA CD, 8A, 26, 34, 00, 8B, 0E, 2C, 00, E8, 07, 00, 17, 07, 1F, 5D, CA
1500 DATA 0E, 00, 53, 26, 88, 17, 43, 26, 88, 27, 43, 49, 49, 26, 88, 07, 43
1510 DATA 26, 88, 27, 43, E2, F6, 26, 88, 37, 43, 26, 88, 27, 5B, C3, 00, 00
1520 DATA 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 73, 74, 61, 63, 6B
1530 DATA 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20
1540 DATA 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20
1550 DATA 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20
1560 DATA 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73
1570 DATA 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74
1580 DATA 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61
1590 DATA 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63
1600 DATA 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B
1610 DATA 20, 20, 20, 73, 74, 61, 63, 6B, 20, 20, 20, 73, 74, 61, 63, 6B, 20
1620 DATA 20, 20, 06, 33, C0, 50, 26, C6, 06, 01, 00, 27, 33, C0, 8E, C0, 26
1630 DATA C7, 06, F0, 04, 00, 00, 26, C7, 06, F2, 04, 00, 00, BA, 20, 03, CB
1640 DATA -1
```

BBS'S POSTING BYTENET LISTINGS

Australia:

Grayham Smith
12 Brentwood Road
Flinders Park, South Australia 5025
The Electronic Oracle
300 Baud, CCITT Standard
Telephone: 08-43-3331 Voice
08-260-6686 BBS

Edward A. Romer
31 Warwick Street
Killara,
Sydney NSW, Australia 2071
OMEN
300 & 1200 Baud
Telephone: 02-498-2399
Voice (Work)
02-499-2642 Voice (Home)
02-498-2495 BBS

Alan Salmon
PCUG Sysop
GPO Box 2229
Canberra,
A.C.T. 2601, Australia
Canberra PC Users Group Inc.
300 & 1200 Baud
Telephone: 61-62-58-9967 BBS

Angus S. Bliss
POB 293
Hamilton NSW 2303, Australia
Newcastle Microcomputer Club
300 Baud, CCITT Standard, 8 Bits, 1 Stop, No Parity
Telephone: 049-67-2433 Voice (Angus Bliss)
049-54-9505 Voice (Tony Nicholson)
61-49-685385 BBS

John Hastwell-Batten
POB 242
Dural, NSW 2158, Australia
Tesseract RCPM+
300 Baud, CCITT Standard, 8 Bits, No Parity
Telephone: 02-651-2363 Voice
02-651-1404 BBS

Phil Harding
POB 35
Charnwood A.C.T., Australia 2615
PC-Exchange Bulletin Board
300 & 1200 Baud, CCITT Standard
Telephone: 61-062-581406 Voice
61-62-586352 BBS

Eric Salter
POB 60
Canterbury 3126, Australia
MICOM: The Microcomputer Club of Melbourne
300 Baud
Telephone: 61-3-861-9117 Eric Salter
61-3-762-1386 Peter Jetson (SYSOP)
61-3-762-5088 BBS

Craig Bowen
29 Warrigal Road
Surrey Hills 3127, Vic., Australia
Public Resource #1
300 Baud, CCITT Standard, 8 Bits, 1 Stop, No Parity
Telephone: 03-890-2174

John Blackett-Smith
Unit 8
69 Wattle Road
Hawthorn 3122, Australia
The National Fido
Telephone: 613-818-2336

Austria:

Wolfgang Hryzak
Bahnstrasse 48
A-2230 Ganserndorf, Austria
University of Vienna BBS FIDO
300 Baud, 8 Bits, 1 Stop Bit
Telephone: 02282-24094 BBS

Brazil:

Sistema Sampa
ATTN: Rizieri Maglio
R. Portugal, 202
Jdm Europe - CEP 01446
Sao Paulo - SP - Brazil
Sistema Sampa
300 & 1200 Baud, CCITT Standard
Telephone: 011-8536273 BBS

Canada:

Leigh Calnek
3036 25th Avenue
Regina, Saskatchewan, Canada S4S 1K9
Telephone: 306-586-9253 BBS

Tom Kashuba
PCOMM Systems
1411 Fort Street, Suite 2001
Montreal, Quebec, Canada H3H 2N7
Telephone: 514-989-9450 BBS

Gary McCallum
Western Canadian Distribution Center
3420 48th Street
Edmonton, Alberta, Canada T6L 3R5
300 & 1200 & 2400 Baud
Telephone: 403-462-9189 Voice
403-461-9124 BBS

Judson Newell
Canada Remote Systems
Suite 311, 4198 Dundas Street West
Toronto, Ontario, Canada M8X 1Y6
Telephone: 416-231-2383 Voice
416-231-9202 BYTEnet System

Vernon Paige
EPSNLINK
3 McNicoll Avenue
Willowdale, Ontario, Canada M2H 2A6
300 & 1200 Baud
Telephone: 416-494-1380 Voice
416-635-9600 BBS

Terry Smythe
Sysop, Z-Node 40
Muddy Water User Group
55 Rowand Avenue
Winnipeg, Manitoba, Canada R3J 2N6
Telephone: 204-832-3982 Voice
204-945-6713 Voice
204-832-4593 BBS

Chile:

Enrique Benavides Z.
Guillermo Gomara C.
Eliodoro Yanez 2210
Providencia
Santiago de Chile
BIGSA BBS
2400/1200/300 Bell & CCITT
Telephone: 562-749848
10:00 am to 4:00 am (Chilean time)
Downloads & Uploads
Xmodem/ASCII

Denmark:

Beverly Kleiman
International Representative
Personal Computer Society of Denmark
Kronprinsensgade 14,
DK-1114 Copenhagen, Denmark
300 Baud, CCITT Standard
Telephone: 01-122518 BBS

England:

Frank Thornley
67 Woodbridge Road,
Guildford,
Surrey GU21 1JP, United Kingdom
CompuLink
Telephone: 0-483-65895 Voice
0-483-573337 (300/1200 Baud) BBS
0-483-573338 (1200/2400 Baud) BBS

Finland:

Juha Wiio
Databox Oy
Museokatu 11
00100 Helsinki, Finland
DATABOX FIDO
300 & 1200 Baud
Telephone: 358-0-497904

Vivian Ronald Dwight
Suvikuja 3 B 14
02120 Espoo, Finland
Micro Maniacs III Fido Node 17
300 & 1200 & 2400 Baud
Telephone: 358-0-424524 Voice
358-0-4557307 Voice
358-0-467673 BBS

France:

Bill Graham,
President
OUF! (Ordinateurs Utilisateurs France)
ATTN: OUFLOG, B.P. 62
10 rue Saint Nicolas
75012 Paris, France
300 Baud, CCITT Standard
Telephone: 331-43-44-06-48 Voice (Bill)
331-43-44-82-65 Voice
331-43-41-61-47 OUFLOG
for BYTE Listings
331-43-40-33-79 OUFTEL
300 & 1200 Baud
331-43-07-95-39 OUFTEL

Dr. Bernard Pidoux
Groupe Des Utilisateurs Francophones D'Informatique
37, Boulevard Saint-Jacques
75014 Paris, France
300 Baud, CCITT Standard
Telephone: 1-47-63-72-50 Voice
1-45-65-10-09 GUFINET
1-45-65-10-11 GUFITEL

Hong Kong:

W. A. Hanafi
SEAnet
Suite 812, Star House,
Tsim Sha Tsui, Kowloon, Hong Kong
ATTN: Christine Wong
Telephone: 5-455088 Voice
5-8937856 SEAnet 1
5-724495 SEAnet 2

continued

Indonesia:

James D. Filgo
US Embassy Box R
APO SF 96356-5000
Jakarta Computer Society
300 Baud, Bell & CCITT Standard
Telephone: 062-21-799-3286 BBS

Israel:

Zohar Levitan
POB 10279
Tshala 61102
Israel
(Contact for telephone number)

Ireland:

Gerry Clarke
30 Auburn Road
Dunlaoire County, Dublin, Ireland
Dublin Bay Bulletin Board
300 & 1200 Baud
Telephone: 353-01-854179

Italy:

Bruno Bonino
MICRO design s.r.l.
Via Rostan, 1
16155 Genova, Italy
C.B.B.S.
CCITT & Bell Standard
Telephone: 10-687098 Voice
10-688783 BBS

Giorgio Leo Rutigliano
Via degli Oleandri, 7
POB 175
85100-Potenza, Italy
FIDO-PZ
300 Baud
Telephone: 0971-34593 Voice (Work)
0971-54431 Voice (Home)
0971-35447 BBS

Claudio Vandelli
Amministratore Unico
SOFT SERVICE s.r.l.
Via G. B. Morgagni 32
20129 Milano, Italy
SOFT SERVICE BBS
300 Baud, CCITT Standard, 8 Bits, No Parity, Full Duplex
Telephone: 02-209231 Voice
02-228467 BBS

Paolo Marraffa
Computronix
Via De Amicis 76
90145 Palermo, Italy
Network Computer Club
300 Baud, CCITT Standard, 8 Bits, 1 Stop Bit, Full Duplex
Telephone: 39-91-266021 BBS
39-91-300229 BBS

Japan:

Peter Perkins
Vice President
Honda Trading Company Ltd.
Mail 101
9-91-Chome, Sota Kanda
Chiyoda-ku, Tokyo, Japan
JANIS
300 & 1200 Baud, CCITT Standard
Telephone: 03-251-0855 BBS

Malaysia:

Ong Boo Huat
3, Jalan Pisang
Jalan Kelang Lama, 58000 Kuala Lumpur
STARLINK
300 Baud
Telephone: 03-7578811 X 116 Voice
03-7576644 BBS

Nigeria:

Chester W. Vlaun
MTCE/31
POB 263
Port Harcourt, Nigeria, West Africa
300 Baud
Telephone: 234-84-301210 to 301229-3022

Norway:

Robert Hertz
Hertz Data Inc.
Huitssfeldts Gate 16
N-0253 Oslo, Norway
Hacker's Unlimited
Telephone: 47-2-431655 Voice
47-2-390521 BBS

Helge Vindenes
5670 FUSA, Norway
Costa del
Telephone: 47-5-151610 Voice
47-5-234129 BBS

Saudi Arabia:

Larry Layland
System Operator DPCS
Aramco
Box 10063
Dhahran, Saudi Arabia 31311
Dhahran Personal Computing Society Bulletin Board
Telephone: 03-873-7851 BBS

Singapore:

Ken Ong
10 Orange Grove Road
#04-01
Singapore 1025, Singapore
K.B.B.S.
300 & 1200 Baud
Telephone: (IDD) 65-734-5825 Voice
(IDD) 65-737-4090 BBS

Sweden:

Jacob Palme
Stockholm University Computer Centre-QZ
Box 27322
102 54 Stockholm, Sweden
BYTECOM
Telephone: 46-8-65-45-00 Voice (Work)
08-23-86-60 (300 Baud)
08-23-89-30 (300 Baud)
08-15-59-20 (300 Baud)
08-14-35-00 (1200 Baud)
08-22-81-30 (1200 Baud)
08-24-61-20 (1200 Baud)
08-14-53-70 (1200 Baud)

Carl Nordin
Nyakersgatan 8B
531 41 Lidkoping, Sweden
A.T.L.
300 & 1200 Baud, CCITT Standard
Telephone: 46-510-25280 Voice
46-510-20409 BBS

Switzerland:

Peter M. C. Werner
9, rue de la Colombiere
1260 Nyon, Switzerland
OCTET
300 & 1200 & 2400 Baud, CCITT Standard
Telephone: 41-22-62-16-54 Voice
41-22-62-18-17 BBS

Albert F. Studer
Technical Director
Kupfer Electronic AG
Soodstrasse 53
Postfach, 8134 Adliswil, Switzerland
TRAX
300 Baud, CCITT Standard
Telephone: 01-710-81-11 Voice
01-710-44-36 BBS

The Netherlands:

Henk Wevers
Cloeckendaal 38
6715 GH Ede, The Netherlands
Henk Wevers' Fido
Telephone: 31-8380-37156 BBS

West Germany:

Rupert Mohr
RMI Nachrichtentechnik GmbH
RosstraBe 7
Postfach 1526
D-5100 Aachen, West Germany
RMI Net
Telephone: 49-241-21145 Voice
45-2410-90528 BYTEnet - DATEX-P
0-26245-2410-90528 User Data - DATEX-P

Rudolf Stricker
Unsoeldstr. 20
D-8000 Munich 22, West Germany
T-BUS FIDO
Telephone: 089-29-38-81 BBS

Announcing BYTE's New Subscriber Benefits Program

Your BYTE subscription brings you a complete diet of the latest in microcomputer technology every 30 days. The kind of broad-based objective coverage you read in every issue. *In addition*, your subscription carries a wealth of other benefits. Check the check list:

DISCOUNTS

- ☑ 13 issues instead of 12 if you send payment with subscription order.
- ☑ One-year subscription at $21 (50% off cover price).
- ☑ Two-year subscription at $38.
- ☑ Three-year subscription at $55.
- ☑ One-year GROUP subscription for ten or more at $17.50 each. (Call or write for details.)

SERVICES

- ☑ *BIX:* BYTE's Information Exchange puts you on-line 24 hours a day with your peers via computer conferencing and electronic mail. All you need to sign up is a microcomputer, a modem, and telecomm software.
- ☑ *Reader Service:* For information on products advertised in BYTE, circle the numbers on the Reader Service card enclosed in each issue that correspond to the numbers for the advertisers you select. Drop it in the mail and we'll get your inquiries to the advertisers.
- ☑ *TIPS:* BYTE's Telephone Inquiry System is available to subscribers who need *fast response*. After obtaining your Subscriber I.D. Card, dial TIPS and enter your inquiries. You'll save as much as ten days over the response to Reader Service cards.

- ☑ *Disks and Downloads:* Listings of programs that accompany BYTE articles are now available free on the BYTEnet bulletin board, and on disk or in quarterly printed supplements.
- ☑ *Microform:* BYTE is available in microform from University Microfilm International in the U.S. and Europe.
- ☑ *BYTE's BOMB:* BYTE's Ongoing Monitor Box is your direct line to the editor's desk. Each month, you can rate the articles via the Reader Service card. Your feedback helps us keep up to date on your information needs.
- ☑ *Customer Service:* If you have a problem with, or a question about, your subscription, you may phone us during regular business hours (Eastern time) at our toll-free number: 800-258-5485. You can also use Customer Service to obtain back issues and editorial indexes.

BONUSES

- ☑ *Annual Separate Issues:* In addition to BYTE's 12 monthly issues, subscribers also receive our annual IBM PC issue free of charge, as well as any other annual issues BYTE may produce.
- ☑ *BYTE Deck:* Subscribers receive five BYTE postcard deck mailings each year—a direct response system for you to obtain information on advertised products through return mail.

To be on the leading edge of microcomputer technology *and* receive all the aforementioned benefits, make a career decision today. Call toll-free weekdays, 8:30am to 4:30pm Eastern time: 800-258-5485.

And . . . welcome to BYTE country!

BYTE
THE SMALL SYSTEMS JOURNAL

McGraw Hill

New! Introducing Turbo C 1.5—the best optimizing compiler gets even better!

The professional optimizing compiler for less than $100

Turbo C® is a technically superior production-quality compiler. (Borland's equation solver, Eureka™, is written in Turbo C.) And our Turbo C 1.5 offers a new library of the highest presentation-quality graphics in the industry—the kind you'll see in Quattro,™ our new professional spreadsheet.

And spectacular graphics are just part of the brand-new features. Turbo C 1.5 enhancements also include:

- A professional-quality graphics library of over 70 functions
- A librarian that allows you to build your own object module libraries
- Context-sensitive help for the language and the library routines

Actual photograph of Turbo C graphics displayed on IBM 8514 screen.*

- Text/video functions, including windows
- 43- and 50-line mode support
- VGA, CGA, EGA, Hercules, and IBM 8514 support
- File search utility (GREP)
- Sample graphics applications
- More than 100 new functions

For professional-quality C at an affordable price, no one else comes close to Turbo C. Because no one can deliver technical superiority like Borland.

*60-Day Money-back Guarantee***

For the dealer nearest you or to order, call

(800) 543-7543

BORLAND INTERNATIONAL

BORLAND

Minimum system requirements: For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles. PC-DOS (MS-DOS®) 2.0 or later. 384K.

*Artwork metafile courtesy of Genigraphics® Corporation

**Customer satisfaction is our main concern; if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

BI 11658

It's easy to upgrade to Turbo C 1.5!

Just complete this coupon and mail it with payment before June 30, 1988. Or, call us at (800) 543-7543 and be ready to give our operators your name, credit card number, and the serial number on your Turbo C master disk.

Turbo C 1.5 Upgrade Price ***$ 33.50***

CA and MA residents add sales tax _____

Shipping and handling
In US $5.00 (Outside US add $10) _____

Total amount enclosed $ _____

Must include your Turbo C serial # _____

Return this coupon and the Turbo C RTL source code registration form from your Turbo C manual along with your payment by March 31, 1988 and receive your Turbo C 1.5 upgrade for free! (No phone orders please.)

Turbo C 1.5 Runtime Library Source Code ***$ 150.00***

CA & MA residents add sales tax _____

Price includes shipping to all US cities.
(Outside US add $10) _____

Total amount enclosed $ _____

Please specify diskette size ☐ 5¼" ☐ 3½"

Method of Payment: ☐ VISA ☐ MC ☐ Check ☐ Bank Draft

Credit card expiration date: _____ / _____

Card # |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Name _____

Ship Address _____

City _____ State _____

Zip _____ Phone (___) _____

Mail coupon to: Turbo C 1.5 Upgrade Dept., Borland International
4585 Scotts Valley Drive, Scotts Valley, CA 95066

This offer is limited to one upgrade per valid product serial number. Not good with any other offer from Borland. Outside US make payments by bank draft payable in US dollars drawn on a US bank. CODs and purchase orders will not be accepted by Borland.